FLOW COMPUTATIONS ON IMPRECISE TERRAINS

Anne Driemel,[*] *Herman Haverkort,*[†] *Maarten Löffler,*[‡] *Rodrigo I. Silveira*[§]

Abstract. We study water flow computation on imprecise terrains. We consider two approaches to modeling flow on a terrain: one where water flows across the surface of a polyhedral terrain in the direction of steepest descent, and one where water only flows along the edges of a predefined graph, for example a grid or a triangulation. In both cases each vertex has an imprecise elevation, given by an interval of possible values, while its (x, y)-coordinates are fixed. For the first model, we show that the problem of deciding whether one vertex may be contained in the watershed of another is NP-hard. In contrast, for the second model we give a simple $O(n \log n)$ time algorithm to compute the minimal and the maximal watershed of a vertex, or a set of vertices, where n is the number of edges of the graph. On a grid model, we can compute the same in $O(n)$ time.

Rose knew almost everything that water can do,
there are an awful lot when you think what.

Gertrude Stein, *The World is Round.*

1 Introduction

Simulating the flow of water on a terrain is a problem that has been studied for a long time in geographic information science (GIS), and has received considerable attention from the computational geometry community due to the underlying geometric problems [7, 8, 21]. It can be an important tool in analyzing flash floods for risk management [2], for stream flow forecasting [18], in the general study of geomorphological processes [5], and it could contribute to obtaining more reliable climate change predictions [27].

When modeling the flow of water across a terrain, it is generally assumed that water flows downward in the direction of steepest descent. It is common practice to compute drainage networks and catchment areas directly from a digital elevation model of the terrain. Most hydrological research in GIS models the terrain surface with a grid in which each cell can drain to one or more of its eight neighbors (e.g. [26]). This can also be modeled as a computation on a graph, in which each node represents a grid cell and each edge

[*] *Dept. of Information and Computing Sciences, Utrecht Univ., Netherlands*, anne@cs.uu.nl. A.D. is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.065.823.

[†] *Dept. of Mathematics and Computer Science, Eindhoven Univ. of Technology, Netherlands*, cs.herman@haverkort.net.

[‡] *Dept. of Information and Computing Sciences, Utrecht Univ., Netherlands*, m.loffler@uu.nl.M.L. was supported by the U.S. Office of Naval Research under grant N00014-08-1-1015, and by the Netherlands Organisation for Scientific Research (NWO) under grant 639.021.123.

[§] *Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain*, rodrigo.silveira@upc.edu. R.S. is supported by the FP7 Marie Curie Actions Individual Fellowship PIEF-GA-2009-251235.

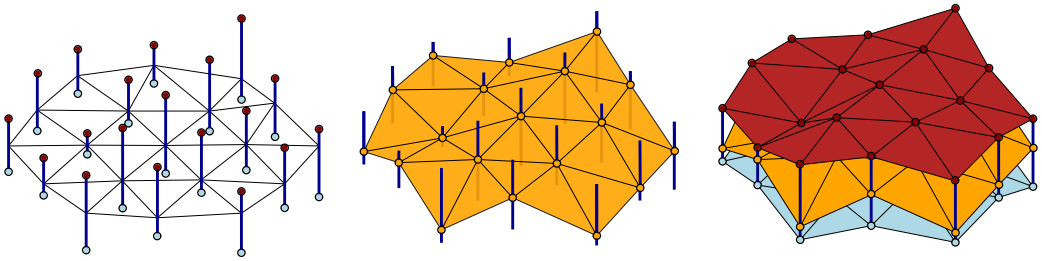


Figure 1: Left: An imprecise terrain. Each vertex of the triangulation has a elevation interval (gray). Center: a realization of the imprecise terrain. Right: the same realization together with the *highest* and *lowest* possible realizations of the imprecise terrain.

represents the adjacency of two neighbors in the grid. Alternatively, one could use an irregular network in which each node drains to one or more of its neighbors, which may reduce the required storage space by allowing less interesting parts of the terrain to have a lower sampling density. We will refer to this as the *network model*, and we assume that, from every node, water flows down along the steepest incident edge. Assuming the elevation data is exact, drainage networks can be computed efficiently in this model (e.g. [6]). In computational geometry and topology, researchers have studied flow path and drainage network computations on triangulated polyhedral surfaces (e.g. [8, 9, 20]). In this model, which we call the *surface model*, the flow of water can be traced across the surface of a triangle. This avoids creating certain artifacts that arise when working with grid models. However, the computations on polyhedral surfaces may be significantly more difficult than on network models [10].

Naturally, all computations based on terrain data are subject to uncertainty, which comes from various sources like measurement, interpolation, and numerical errors. The GIS community has long recognized the importance of dealing with uncertainty explicitly, in particular for hydrological modeling [1, 29]. A natural approach is to model the elevation at a point of the terrain using stochastic methods. However, the models available in the hydrology literature are unsatisfactory [3, 22, 25] and computationally expensive [28]. A particular challenge is posed by the fact that hydrological computations can be extremely sensitive to small elevation errors [15, 19]. While most of these studies have been done in the network model, we note that there also exists work on the behavior of watersheds on noisy terrains in the surface model [14].

A non-probabilistic model of imprecision that is often used in computational geometry consists in representing an imprecise attribute (such as location) by a region that is guaranteed to contain the true value. This approach has also been applied to polyhedral terrains (e.g. [13, 17]), replacing the exact elevation of each surface point by an imprecision interval (see Figure 1). In this way, each terrain vertex does not have one fixed elevation, but a whole range of possible elevations which includes the true elevation. Choosing a concrete elevation for each vertex results in a *realization* of the imprecise terrain. The realization is a (precise) polyhedral terrain. Since the set of all possible realizations is guaranteed to include the true (unknown) terrain, one can now obtain bounds on parameters of the true terrain by computing the best- and worst-case values of these parameters over the set of all

possible realizations. Note that we assume that there is only an error in the z-coordinate (and not in the x, y-coordinates). This is partially motivated by the fact that commercial terrain data suppliers often only report elevation error [11]. However, it is also a natural simplification of the model, since the true terrain needs to have an elevation at any exact position in the plane.

In this paper we apply this model of imprecise terrains to problems related to the simulation of water flow, both on terrains represented by surface models and on terrains represented by network models. One of the most fundamental questions one may ask about water flow on terrains is whether water flows from a point p to another point q. In the context of imprecise terrains, reasonable answers may be "definitely not", "possibly", and "definitely". The *watershed* of a point in a terrain is the part of the terrain that drains to this point. Phrasing the same question in terms of watersheds leads us to introduce the concepts of *potential* (maximal) and *persistent* (minimal) watersheds.

Results In Section 3 we show that the problem of deciding whether water can flow between two given points in the surface model is NP-hard. Fortunately, the situation is much better for the network model, and therefore as a special case also for the D-8 grid model which is widely adopted in GIS applications. We present various results using this model in Section 4 and Section 5. In Section 4.1 we present an algorithm to compute the potential watershed of a point. On a terrain with n edges, our algorithm runs in $O(n \log n)$ time; for grid models the running time can even be improved to $O(n)$. We extend these techniques and achieve the same running times for computing the potential downstream area of a point in Section 4.2 and its persistent watershed in Section 4.3. In order to be able to extend these results in the network model, we define a certain class of imprecise terrains which we call *regular* in Section 5.1. We discuss an algorithm that turns a non-regular terrain into a regular one in Section 5.2. We prove that persistent watersheds satisfy certain nesting conditions on regular terrains in Section 5.3. This leads to efficient computations of fuzzy watershed boundaries in Section 5.4, and to the definition of the *fuzzy ridge* in Section 5.5, which delineates the persistent watersheds of the "main" minima of a regular terrain and which is equal to the union of the areas where the potential watersheds of these minima overlap. We can compute this structure in $O(n \log n)$ time (see Theorem 6). We conclude the paper with a discussion of open problems in Section 6.

2 Preliminaries

In this section we give the definition of imprecise terrains and realizations and discuss the two flow models used in this paper.

2.1 Basic definitions and notation

We define an ***imprecise terrain*** T as a possibly non-planar geometric graph G with nodes $V \subset \mathbb{R}^2$ and edges $E \subseteq V \times V$, where each node $v \in V$ has an imprecise third coordinate, which represents its ***elevation***. We denote the bounds of the elevation of v with $low(v)$ and

$high(v)$. A ***realization*** R of an imprecise terrain T consists of the given graph together with an assignment of elevations to nodes, such that for each node v its elevation $elev_R(v)$ is at least $low(v)$ and at most $high(v)$. We denote with R^- the realization such that $elev_{R^-}(v) = low(v)$ for every node v and, similarly, we denote with R^+ the realization such that $elev_{R^+}(v) = high(v)$. The set of all realizations of an imprecise terrain T is denoted $\mathcal{R}_T$.

For any set of nodes $P \subseteq V$, we define the ***neighborhood*** of P as the set of nodes $N(P) = \{s : s \notin P \wedge \exists\, t \in P : (s,t) \in E\}$. Now, consider a realization R of an imprecise terrain as defined above. A set of nodes $P \subseteq V$ constitutes a ***local minimum*** in R if the following conditions are met: (i) the subgraph of G induced by P is connected, (ii) all nodes of P have the same elevation according to R, and (iii) their elevation is strictly lower than the elevation of any node in $N(P)$ according to R. Likewise, a local maximum is a set of nodes at the same elevation of which the neighborhood is strictly lower.

2.2 A model of discrete water flow

Consider a realization R of an imprecise terrain as defined above. If water is only allowed to flow along the edges of the realization, then the realization represents a network. Therefore we refer to this model of water flow as the ***network model***. Below, we state more precisely how water flows in this model and give a proper definition of the watershed. This model or variations of it have been used before, for example in [6, 23, 26].

The steepness of descent (slope) of an edge $(p,q) \in E$ in realization R is defined as $\sigma_R(p,q) = (elev_R(p) - elev_R(q))/|pq|$, where $|pq|$ is the Euclidean distance between p and q. The node q is a ***steepest-descent neighbor*** of p in R, if and only if $\sigma_R(p,q)$ is non-negative and maximal over all neighbors q of p. In the realization R, water that arrives in p will continue to flow to each of its steepest-descent neighbors, unless p constitutes a local minimum. If there exists a local minimum $P \ni p$, then the water that arrives in p will flow to the neighbors of p in P and eventually reach all the nodes of P, but it will not flow further to any node outside the set P. If water from p reaches a node $q \in V$ then we write $p \overset{R}{\rightarrow} q$ ("p flows to q in R"), and for technical reasons we define $p \overset{R}{\rightarrow} p$ for all p and R.

The ***discrete watershed*** of a node q in a realization R is defined as the union of nodes that flow to q in R, that is $\mathcal{W}_R(q) := \{p \ : \ p \overset{R}{\rightarrow} q\}$. Similarly, we define the discrete watershed of a set of nodes Q in this realization as $\mathcal{W}_R(Q) := \bigcup_{q \in Q} \mathcal{W}_R(q)$.

Consider the graph G of the imprecise terrain. A path π in G is a ***flow path*** for a realization R if it does not self-intersect (any node appears on the path at most once) and each node on the path (except the first) is a steepest-descent neighbor of its predecessor on the path. For any pair of nodes p, q in π, we write $p \overset{\pi}{\rightarrow} q$ if π contains p and q in this order. For any set of realizations $\mathcal{S} \subseteq \mathcal{R}_T$, we denote with $\Pi^*(\mathcal{S})$ the set union of all flow paths induced by a realization in $\mathcal{S}$. We define a ***maximal flow path*** as a flow path that ends in a local minimum and cannot continue without intersecting itself.

2.2.1 Flow paths are stable

This subsection is a note on flow paths, which we defined for the network model above. We define when a flow path is stable and argue that any flow path induced by a realization in $\mathcal{R}_T$ is stable with respect to some ε-neighborhood of $\mathcal{R}_T$. Intuitively, the analysis in this section shows that the flow paths considered in our model are never the result of an isolated degenerate situation, but could also exist if the estimated elevation intervals of the vertices would be slightly different. This may serve as a justification or proof of soundness of the network model.

For two realizations $R, R' \in \mathcal{R}_T$, we call R' an ***ε-perturbation*** of R if for all nodes $v \in V$ it holds that $|\, elev_R(v) - elev_{R'}(v)| \leq \varepsilon$. For a set of realizations $\mathcal{S}$, let $\mathcal{S}^\varepsilon$ denote the union of $\mathcal{S}$ with the ε-perturbations of elements of $\mathcal{S}$. We say that a flow path π is ***stable*** with respect to $\mathcal{S}$ if for some $\varepsilon > 0$ the flow path exists in any ε-perturbation of some $R \in \mathcal{S}$. In this context, we call R a ***perturbation center*** of π.

Lemma 1. *Given a set of realizations $\mathcal{S}$ and any value $\delta > 0$, it holds that any flow path π induced by a realization in $\mathcal{S}$ is stable with respect to $\mathcal{S}^\delta$.*

Proof. We call a realization which does not contain horizontal edges and in which any node has at most one steepest-descent neighbor ***non-ambiguous***, similarly, a realization for which any of these properties does not hold is called ambiguous. Any flow path π induced by a non-ambiguous realization $R \in \mathcal{S}$ is stable with perturbation center R, since we can pick ε small enough such that the order of the slopes of the edges does not change. Now, let $\pi = p_1, p_2, \ldots, p_k$ be a flow path from p_1 to p_k which is induced by an ambiguous realization $R \in \mathcal{S}$. We lower each node p_i by $\delta/2+(i\delta)/(4k)$ and perturb the remaining vertices by some value smaller than $\varepsilon/4$. Since π is non-intersecting, we create a non-ambiguous realization $R' \in \mathcal{S}^\delta$ in this way which also induces π. This proves the claim. □

2.3 A model of continuous water flow

Consider an imprecise terrain, where the graph that represents the terrain forms a planar triangulation in the (x, y)-domain. Any realization of this terrain is a polyhedral terrain with a triangulated surface. If we assume that the water which arrives at a particular point p on this surface will always flow in the true direction of steepest descent at p across the surface, possibly across the interior of a triangle, then we obtain a continuous model of water flow. Since the steepest-descent paths do not necessarily follow along the edges of the graph, but instead lead across the surface formed by the graph, we call this model the ***surface model***. This model has also been used before, for example in [8, 9, 20].

Since, as we will show in the next section, it is already NP-hard to decide whether water from a point p can potentially flow to another point q, we will focus on the network model in the rest of the paper, and we do not formally define watersheds in the surface model. Therefore, we will simply use the term "watershed" to refer to discrete watersheds in this paper.

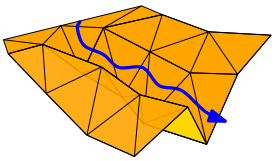
3 NP-hardness in the surface model

In the surface model water flows across the surface of a polyhedral terrain; refer to Section 2.3 for the details of the model. In this section we prove that it is NP-hard to decide whether water potentially flows from a point s to another point t in this model. The reduction is from 3-SAT; the input is a 3-CNF formula with n variables and m clauses. We first present the general idea of the proof, then we proceed with a detailed description of the construction, and finally we prove the correctness.

3.1 Overview of the construction

The main idea of the NP-hardness construction is to encode the variables and clauses of the 3-SAT instance in an imprecise terrain, such that a truth assignment to the variables corresponds to a realization—i.e., an assignment of elevations—of this terrain. If and only if all clauses are satisfied, water will flow from a certain starting vertex s to a certain target vertex t. We first introduce the basic elements of the construction: channels and switch gadgets.

Channels We can mold channels in the fixed part of the terrain to route water along any path, as long as the path is monotone in the direction of steepest descent on the terrain. We do this by increasing the elevations of vertices next to the path, thus building walls that force the water to stay in the channel. We can end a channel in a local minimum anywhere on the fixed part of the terrain, if needed.



Switch gadgets The general idea of a *switch gadget* is that it provides a way for water to switch between channels. A simple switch gadget has one incoming channel, three outgoing channels, and two *control vertices* a and b, placed on the boundary of the switch. The water from the incoming channel has to flow across a central triangle, which is connected to a and b. Depending on their elevations, the two vertices a and b divert the water from the incoming channel to a particular outgoing channel and thereby "control" the behavior of the switch gadget. This is possible, since the slopes of the central triangles, which the water needs to pass, depend on the elevations of a and b and those two are the only vertices with imprecise elevations. The elevations of the remaining vertices which define the gadget are fixed. Refer to Figure 2 for an illustration.

We can also build switches for multiple incoming channels. In this case, every incoming channel has its own dedicated set of outgoing channels, and it is also controlled by only two vertices, see Figure 3. Note that we can lead the middle outgoing channel to a local minimum as shown in the examples and, in this way, ensure that, if any water can pass the switch, the elevations of its control vertices are at unambiguous extremal elevations. Depending on the particular construction of the switch, we may want them to be at opposite extremal elevations or at corresponding extremal elevations.

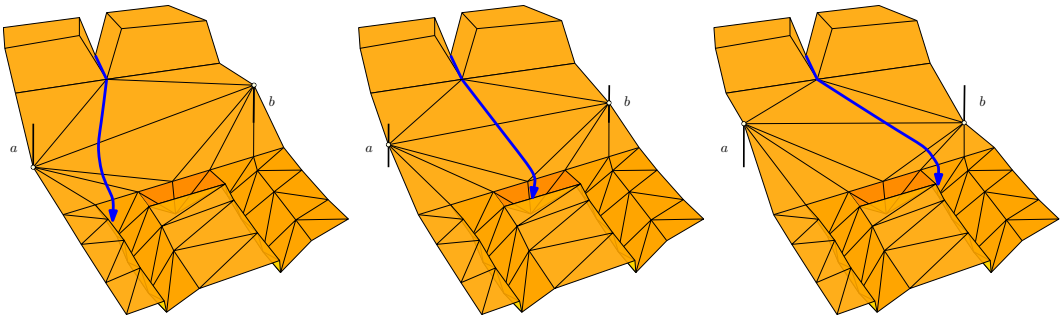


Figure 2: Three different states of a simplistic switch gadget.

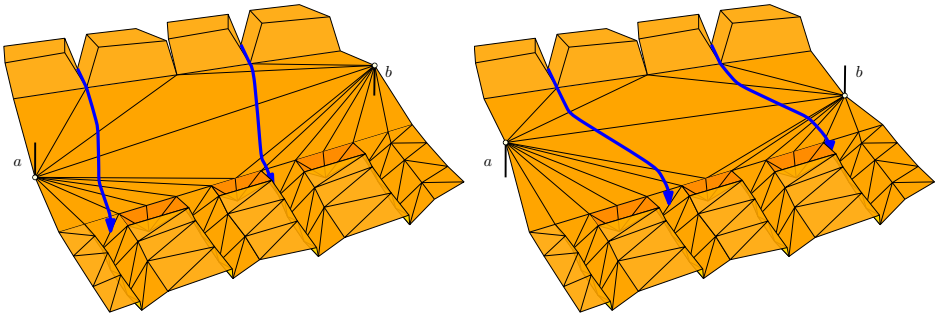


Figure 3: Sketch of a switch with multiple incoming channels.

Global layout The global layout of the construction is depicted in Figure 4. The construction contains a grid of $m \times n$ cells, in which each clause corresponds to a column and each variable to a row of the grid. The grid is placed on the western slope of a "mountain"; columns are oriented north-south and rows are oriented east-west. We create a system of channels that spirals around this mountain, starting from s at the top and ending in t at the bottom of the mountain. We ensure that in no realization, water from s can escape this channel system and, if it reaches t, we know that it followed a strict course that passes through every cell of the grid exactly once, column by column from east to west, and within each column, from north to south. Embedded in this channel system, we place a switch gadget in every cell of the grid, which allows the water from s to "switch" from one channel to another within the current column depending on the elevations of the vertices that control the gadget. In this way, the switch gadgets of a row encode the state of a variable. To ensure that the state of a variable is encoded consistently across a row of the grid, the switch gadgets in a row are linked by their control vertices. Every column has a dedicated entry point at its north end, and a dedicated exit point at its south end. If and only if water flows between these two points, the clause that is encoded in this column is satisfied by the corresponding truth assignment to the variables. The slope of the mountain is such that columns descend towards the south, and the exit point of each column (except the westernmost one) is higher than the entry point of the adjacent column to the west; water can flow between these points through a channel around the back of the mountain. The easternmost column's entry point is the starting vertex s, and the westernmost column's exit point is the target vertex t.

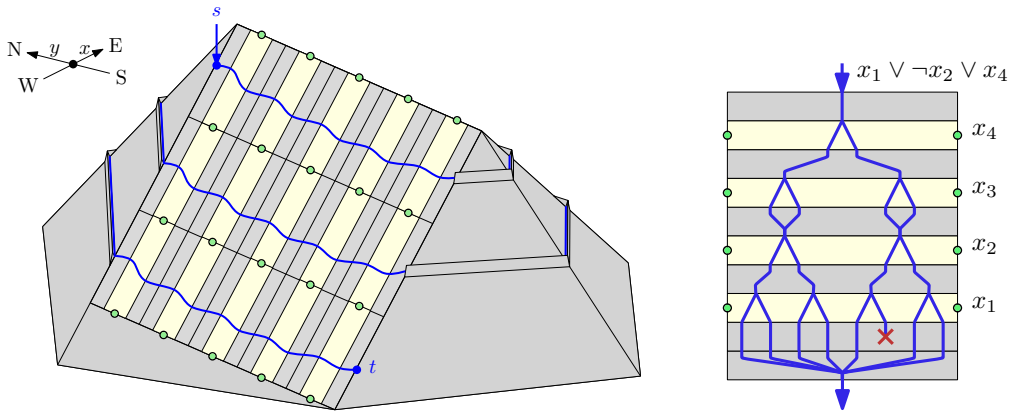


Figure 4: Left: Global view of the NP-hardness construction, showing the grid on the mountain slope. The fixed parts are shown in gray, the variable parts are shown light yellow and the imprecise vertices are filled light green; Right: Detail of a clause, which forms one of the columns of the grid.

Clause columns To encode each clause, we connect the switch gadgets in a column of the global grid by channels in a tree-like manner. By construction, water will arrive in a different channel at the bottom of the column for each of the eight possible combinations of truth values for the variables in the clause. This is possible because a switch gadget can switch multiple channels simultaneously. We let the channel in which water would end up if the clause is not satisfied lead to a local minimum; the other seven channels merge into one channel that leads to the exit point of the clause. The possible courses that water can take will also cross switch gadgets of variables that are not part of the clause: in that case, each course splits into two courses, which are merged again immediately after emerging from the switch gadget. Figure 4 (right) shows an example.

Sloped switch gadgets Since the grid is placed on the western slope of a mountain, water on the central triangle of a switch will veer off towards the west, regardless of the elevations of its control vertices. However, as we will see, we can still design a working switch gadget in this case. Recall that we link the switch gadgets of a variable row by their control vertices, such that each switch gadget shares one control vertex with its neighboring cell to the west and one with its neighboring cell to the east. As mentioned before, such a row encodes the state of a particular variable. We say that it is in a consistent state if either all control vertices of the switches are *high* or all control vertices are *low*. Thus, we will use the following assignment of truth values to the elevations of the control vertices of our switch gadgets: both vertices set to their highest elevation encodes *true*; both vertices set to their lowest elevation encodes *false*; other combinations encode *confused*. Depending on the truth value encoded by the elevations of the imprecise vertices, water that enters the gadget will flow to different channels. The channels in which the water ends up when the gadget reads *confused* always lead to a local minimum. For the other channels, their destination depends on the clause. In Figure 5 you can see a sketch of a sloped switch gadget which works the way described above.

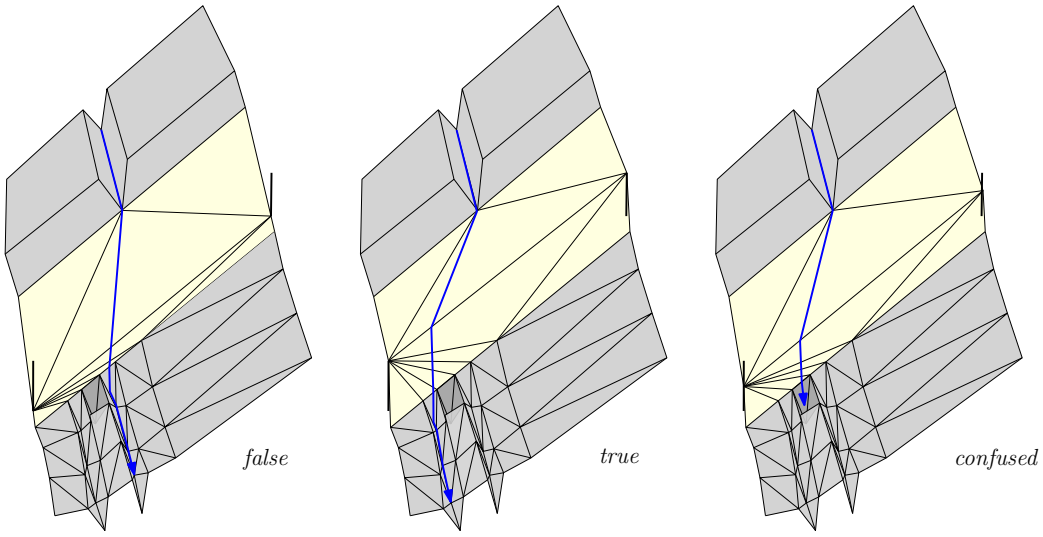


Figure 5: Illustration of a sloped switch gadget similar to the one used in the final construction. The final gadget has multiple incoming channels, which is not shown in this figure.

3.2 Details of the construction

Recall that we are given a 3-SAT instance with n variables and m clauses. The central part of the construction, which will contain the gadgets, consists of a grid of n rows—one for each variable—and m columns—one for each clause. We denote the width of each row, measured from north to south, by $B = 400$, and the width of a column, measured from west to east, by $A = \max((n+1) \cdot B, 4000)$. Ignoring local details, on any line from north to south in this part of the construction, the terrain descends at a rate of $dz/dy = 1$, and on any line from east to west, it descends at a rate of $dz/dx = 1$; thus we have $z = x + y$. Observe that each column measures $nB < A$ from north to south; thus the southern edge of each column is at a higher elevation than the northern edge of the next column to the west. The dedicated entrance and exit points of column $1 \le j \le m$ are placed at $(jA - \frac{1}{2}A, nB, jA - \frac{1}{2}A + nB)$ and $(jA - \frac{1}{2}A, 0, jA - \frac{1}{2}A)$, thus allowing the construction of a descending channel from each column's exit point to the entry point of the column to the west.

For every variable v_i, $1 \le i \le n$, we place $m+1$ imprecise vertices v_{ij}, for $0 \le j \le m$, in row i, on the boundaries of the columns corresponding to the m clauses. Vertex v_{ij} has x-coordinate jA, y-coordinate $iB - \frac{1}{2}B$, and an imprecise z-coordinate $[jA + iB - \frac{1}{2}B$, $jA + iB - \frac{1}{2}B + 20]$. On every pair of imprecise vertices $v_{i(j-1)}, v_{ij}$ we build a switch gadget G_{ij}; thus there is a switch gadget for each variable/clause pair. The coordinates of the vertices in each gadget, relative to the coordinates of $v_{i(j-1)}$, can be found in Figure 6.

Switch gadget construction We use the sloped switch gadget described above and illustrated in Figure 5. Our switch gadget occupies a rectangular area that is A wide from west

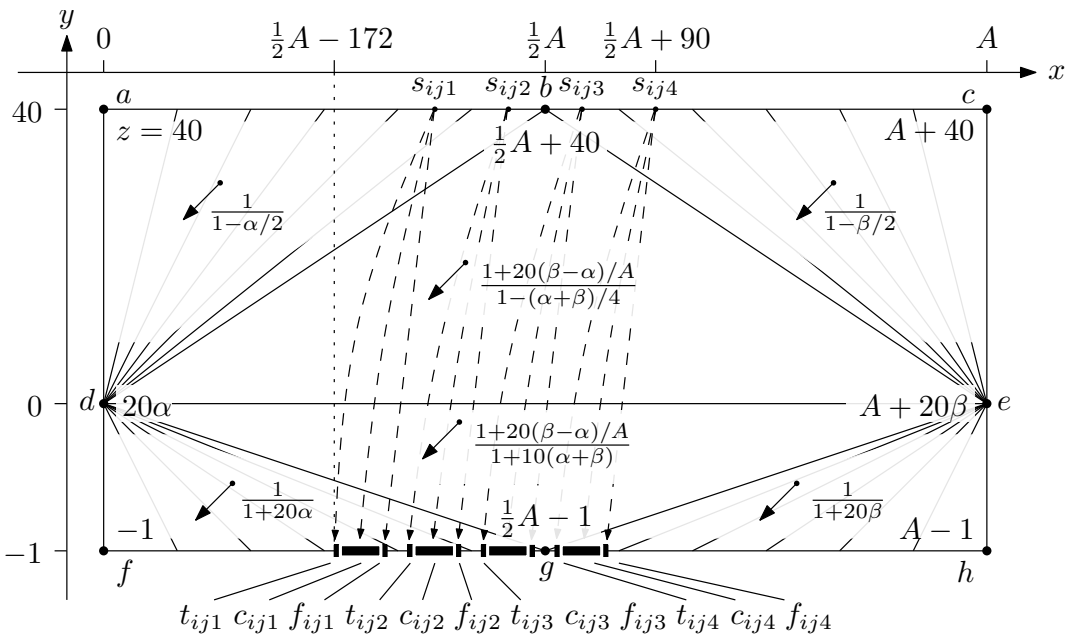


Figure 6: Distances and gradients on a connector gadget. All coordinates are relative to the lowermost position of the control vertex $d = v_{i(j-1)}$. The other control vertex is $e = v_{ij}$. Thus, e and d are the only imprecise vertices. The x- and y-coordinates of the vertices are indicated on the axes. The elevations of the key vertices are written next to the vertices. The elevations of the control vertices are expressed as a function of $\alpha, \beta \in [0, 1]$. The directions of steepest descent on the different faces of the gadget (marked with arrows) are expressed in the form dx/dy, as a function of α and β.

to east, and 41 wide from north to south. Its key vertices and their coordinates, relative to each other, can be found in Figure 6. There are two imprecise vertices, d and e, with elevation range $[0, 20]$ and $[A, A + 20]$, respectively—so in any realization, their elevations have the form 20α and $A + 20\beta$, respectively, where $\alpha, \beta \in [0, 1]$.

On the north edge of the gadget, there may be many more vertices, all collinear with a, b and c. The vertices on the western half of the north edge are connected to the western control vertex, and the vertices on the eastern half of the north edge are connected to the eastern control vertex. In particular, each gadget G_{ij} is designed to receive water from four channels that arrive at four points $s_{ij1}, s_{ij2}, s_{ij3}, s_{ij4}$ on the north edge, close to b; the coordinates of these points are $s_{ijk} = (\frac{1}{2}A - 150 + 60k, 40, \frac{1}{2}A - 110 + 60k)$.

On the south edge of the gadget, there is a similar row of vertices, all collinear with f, g and h, that are connected to the control vertices. To the south, the gadget is connected to twelve channels that catch all water that arrives at certain intervals on the south edge: for each $k \in \{1, 2, 3, 4\}$, there is a *western channel* t_{ijk} catching all water arriving between $s_{ijk} - (82, 41, 123)$ and $s_{ijk} - (77, 41, 118)$, a *middle channel* c_{ijk} catching all water arriving between $s_{ijk} - (77, 41, 118)$ and $s_{ijk} - (44, 41, 85)$, and an *eastern channel* f_{ijk} catching all water arriving between $s_{ijk} - (44, 41, 85)$ and $s_{ijk} - (39, 41, 80)$.

In a particular realization R, we define the switch gadget to be in a *false* state if $\alpha = \beta = 0$, in a *true* state if $\alpha = \beta = 1$, and in a *confused* state if $\alpha \leq \frac{1}{2}$ while $\beta \geq \frac{1}{2}$, or if $\alpha \geq \frac{1}{2}$ while $\beta \leq \frac{1}{2}$. As we will show below, in the true, false, and confused states the gadget leads any water that comes in at any point s_{ijk} into t_{ijk}, f_{ijk}, and c_{ijk}, respectively.

We model the fixed part of the terrain such that the middle channels all lead to local minima. The western and eastern channels correspond to a (partial) truth assignment of the variables of the clause that is represented by the column that contains the gadget; these channels lead to a local minimum or to the next row, as described below.

Constructing the clause columns Each clause is modeled in a column j by making certain connections between the outgoing channels of each gadget to the dedicated entrance points of the gadget in the next row. Observe that by our choice of B, the entrance point of column j lies above all entrance points of G_{nj}, all outgoing channels of any gadget $G_{(i+1)j}$ start at higher elevations than all entrance points of G_{ij}, and all outgoing channels of G_{1j} start at an elevation higher than the exit point of the column. This ensures that all channels described below can indeed be built as monotonously descending channels, so that water can flow through it. We will now explain the connections which we use to build a clause.

Let $p > q > r$ be the indices of the variables that appear in the clause. The water courses modeling the clause start at the entry point of the column, from which any water is led through a channel to entry point s_{nj1} of gadget G_{nj}.

For $i \neq \{p, q, r\}, k \in \{1, 2, 3, 4\}$, we connect both t_{ijk} and f_{ijk} to $s_{(i-1)jk}$ (if $i > 1$) or to the exit point of the column (if $i = 1$).

We connect t_{pj1} and f_{pj1} to $s_{(p-1)j1}$ and $s_{(p-1)j2}$, respectively. Thus, for $i \in \{q, ..., p-1\}$, water that enters G_{ij} at s_{ij1} and s_{ij2} represents the cases that p is true and p is false, respectively.

We connect $t_{qj1}, f_{qj1}, t_{qj2}$ and f_{qj2} to $s_{(q-1)j1}, s_{(q-1)j2}, s_{(q-1)j3}$ and $s_{(q-1)j4}$, respectively. Thus, for $i \in \{r, ..., q-1\}$, water that enters G_{ij} at $s_{ij1}, s_{ij2}, s_{ij3}$ and s_{ij4} represents the four different possible combinations of truth assignments to p and q, respectively.

The eight channels $t_{rj1}, f_{rj1}, t_{rj2}, f_{rj2}, t_{rj3}, f_{rj3}, t_{rj4}, f_{rj4}$ now represent the eight different possible combinations of truth assignments to the variables of the clause. The channel that corresponds to the truth assignment that renders the clause false, is constructed such that it ends in a local minimum. The other seven channels all lead to $s_{(r-1)j1}$ (if $r > 1$) or to the exit point of the clause column (if $r = 1$).

3.3 Analysis of flow through a gadget

Below we will analyze where water may leave a gadget G_{ij} after entering the gadget at point s_{ijk}, with x-coordinate x_k. In the discussion below, all coordinates are relative to the lowermost position of the western control vertex of the gadget—refer to Figure 6, which also shows the directions of steepest descent (i.e. the surface gradients, expressed as dx/dy) on each face of the gadget.

First observe that in any case, the directions of steepest descent on $\triangle abd$, $\triangle bce$ and

$\triangle bed$ are at least $1 - 20/A \geq 199/200 = 0.995$ and at most $(1 + 20/A)/(1/2) \leq 201/100 = 2.01$. Thus, when the water reaches y-coordinate 38, it will be at x-coordinate at least $x_k - 4.02$ and at most $x_k - 1.99$.

Note that the line bd intersects the plane $y = 38$ at $x = \frac{1}{2}A - \frac{1}{40}A \leq \frac{1}{2}A - 100$, and the line be intersects the plane $y = 38$ at $x = \frac{1}{2}A + \frac{1}{40}A \geq \frac{1}{2}A + 100$. By our choice of coordinates for the entrance points s_{ijk}, we have $|x_k - \frac{1}{2}A| \leq 90$; therefore the water will be on $\triangle bed$ when it reaches $y = 38$. Let $g_{\max}$ and $g_{\min}$ be the maximum and minimum possible gradients dx/dy on $\triangle bed$, respectively. Thus, the water will reach the line de at x-coordinate at least $x_k - 4.02 - 38g_{\max}$ and at most $x_k - 1.99 - 38g_{\min}$.

Finally, the directions of steepest descent on $\triangle dgf$, $\triangle egh$ and $\triangle deg$ are more than 0 and less than $1 + 20/A \leq 201/200 < 1.01$. Thus, the water will reach the line fh at x-coordinate more than $x_k - 5.03 - 38g_{\max}$ and less than $x_k - 1.99 - 38g_{\min}$.

We will now consider five classes of configurations of the control vertices in the gadget, and compute the interval of x-coordinates where water may reach the line fh in each case.

- $\alpha = \beta = 1$ *(true state)* In this case we have $g_{\max} = g_{\min} = 2$, so water will reach the line fh within the x-coordinate interval $(x_k - 81.03, x_k - 77.99)$, and thus it will flow into channel t_{ijk}.
- $\alpha + \beta > \frac{3}{2}$ *(true-ish state)* In this case we have $g_{\max} \leq (1 + 20/A)/(1/2) \leq 2.01$ and $g_{\min} \geq (1 - 20/A)/(1 - 3/8) \geq 199/125 > 1.59$. Thus water will reach the line fh within the x-coordinate interval $(x_k - 81.41, x_k - 62.41)$, and thus it will flow into channel t_{ijk} or c_{ijk}.
- $\frac{1}{2} \leq \alpha + \beta \leq \frac{3}{2}$ *(this includes all proper confused states)* In this case we have $g_{\max} \leq (1 + 20/A)/(1 - 3/8) \leq 201/125 < 1.61$ and $g_{\min} \geq (1 - 20/A)/(1 - 1/8) \geq 199/175 > 1.13$. Thus water will reach the line fh within the x-coordinate interval $(x_k - 66.21, x_k - 44.93)$, and thus it will flow into channel c_{ijk}.
- $\alpha + \beta < \frac{1}{2}$ *(false-ish state)* In this case we have $g_{\max} \leq (1 + 20/A)/(1 - 1/8) \leq 201/175 < 1.15$ and $g_{\min} \geq (1 - 20/A) \geq 199/200 > 0.99$. Thus water will reach the line fh within the x-coordinate interval $(x_k - 48.73, x_k - 39.61)$, and thus it will flow into channel c_{ijk} or f_{ijk}.
- $\alpha = \beta = 0$ *(false state)* In this case we have $g_{\max} = g_{\min} = 1$, so water will reach the line fh within the x-coordinate interval $(x_k - 43.03, x_k - 39.99)$, and thus it will flow into channel f_{ijk}.

3.4 Correctness of the NP-hardness reduction

Lemma 2. *If water flows from s to t in some realization, then there is a truth assignment that satisfies the 3-CNF formula.*

Proof. Water that starts flowing from s, which is the entrance point of the clause column m, is immediately forced into a channel to entrance point s_{nm1} of gadget G_{nm}. As calculated

above, any water that enters a gadget at one of its designated entrance points will leave the gadget in one of its designated channels, which leads either to a local minimum, or to a designated entrance point of the next gadget. Therefore, water from s can only reach t after flowing through all switch gadgets.

Since all middle outgoing channels c_{ijk} lead to local minima, we know that if there is a flow path from s to t, then the water from s is nowhere forced into a middle outgoing channel. It follows that no gadget is in a proper confused state. As a consequence, in any row, either all gadgets have their control vertices in the lower open half of their elevation range, or all gadgets have their control vertices in the upper open half of their elevation range. In the first case, all gadgets in the row are in a *false-ish* state, and any incoming water from s leaves those gadgets in the same channels as if the gadgets were in a proper *false* state. In the second case, all gadgets in the row are in a *true-ish* state, and any incoming water from s leaves those gadgets in the same channels as if the gadgets were in a proper *true* state.

We can now construct a truth assignment $\mathcal{A}$ to the variables, in which each variable is *true* if the control vertices in the corresponding row are in the upper halves of their elevation ranges, and *false* otherwise. It follows from the way in which channel networks in clause columns are constructed, that in each clause column, water will flow into one of the seven channels that corresponds to a truth assignment that satisfies the corresponding clause—otherwise the water would not reach t. Therefore, $\mathcal{A}$ satisfies each clause, and thus, the complete 3-CNF formula. □

Lemma 3. *If there is a truth assignment to the variables that satisfies the given 3-CNF formula, then there is a realization of the imprecise terrain in which water flows from s to t.*

Proof. We set all control vertices in rows corresponding to true variables to their highest positions and all control vertices in rows corresponding to false variables to their lowest positions. One may now verify that, by construction, in each clause column water from the column's entry point will flow into one of the seven channels that lead to the column's exit point, and thus, water from s reaches t. □

Thus, 3-SAT can be reduced, in polynomial time, to deciding whether there is a realization of T such that water can flow from s to t. We conclude that deciding whether there exists a realization of T such that water can flow from s to t is NP-hard.

Theorem 1. *Let T be an imprecise triangulated terrain, and let s and t be two points on the terrain. Deciding whether there exists a realization $R \in \mathcal{R}_T$ such that $p \xrightarrow[R]{} q$ is NP-hard.*

4 Watersheds in the network model

In the network model we assume that water flows only along the edges of a realization. More specifically, water that arrives in a node p continues to flow along the steepest-descent edges incident to p, unless p is a local minimum. For a formal definition of the watershed and flow paths please refer to Section 2.2.

4.1 Potential watersheds

The ***potential watershed*** of a set of nodes Q in a terrain T is defined as

$$\mathcal{W}_{\cup}(Q) := \bigcup_{R \in \mathcal{R}_T} \bigcup_{q \in Q} \mathcal{W}_R(q),$$

which is the union of the watersheds of Q over all realizations of T. This is the set of nodes from which there exists a flow path to a node of Q in some realization. With slight abuse of notation, we may also write $\mathcal{W}_{\cup}(q)$ to denote the potential watershed of a single node q.

4.1.1 Canonical realizations

We prove that for any given set of nodes Q in an imprecise terrain, there exists a realization R such that $\mathcal{W}_R(Q) = \mathcal{W}_{\cup}(Q)$. For this we introduce the notion of the overlay of a set of watersheds in different realizations of the terrain. Informally, the overlay is a realization that sets every node that is contained in one of these watersheds to the lowest elevation it has in any of these watersheds.

Definition 1. *Given a sequence of realizations $R_1, ..., R_k$ and a sequence of nodes $q_1, ..., q_k$, the* **watershed-overlay** *of $\mathcal{W}_{R_1}(q_1), ..., \mathcal{W}_{R_k}(q_k)$ is the realization $\overline{R}$ such that for every node v, we have that $elev_{\overline{R}}(v) = high(v)$ if $v \notin \bigcup \mathcal{W}_{R_i}(q_i)$ and otherwise*

$$elev_{\overline{R}}(v) = \min_{i : v \in \mathcal{W}_{R_i}(q_i)} elev_{R_i}(v).$$

Note that we allow ourselves a slight abuse of wording and notation here: the input to the watershed-overlay operation is not a set of watersheds, but a sequence of realizations and a sequence of nodes.

Lemma 4. *Let $\overline{R}$ be the watershed-overlay of $\mathcal{W}_{R_1}(q_1), \ldots, \mathcal{W}_{R_k}(q_k)$, and let $Q = \bigcup_{1 \le i \le k} q_i$, then $\mathcal{W}_{\overline{R}}(Q)$ contains $\mathcal{W}_{R_i}(q_i)$, for any $1 \le i \le k$.*

Proof. Let u be a node of the terrain. We prove the lemma by induction on increasing symbolic elevation to show that if u is contained in one of the given watersheds, then it is also contained in $\mathcal{W}_{\overline{R}}(Q)$. To this end, we define $level(R_i, u)$ as the smallest number of edges on any path along which water flows from u to q_i in R_i; if there is no such path, then $level(R_i, u) = \infty$. Now we define the ***symbolic elevation*** of u, denoted $elev^*(u)$, as follows: if u is contained in any watershed $\mathcal{W}_{R_i}(q_i)$, then $elev^*(u)$ is the lexicographically smallest tuple $(elev_{R_i}(u), level(R_i, u))$ over all i such that $u \in \mathcal{W}_{R_i}(q_i)$; otherwise $elev^*(u) = (high(u), \infty)$.

Now consider a node u that is contained in one of the given watersheds. The base case is that u is contained in Q, and in this case the claim holds trivially. Otherwise, let R_i be a realization such that $u \in \mathcal{W}_{R_i}(q_i)$ and such that $(elev_{R_i}(u), level(R_i, u))$ is lexicographically smallest over all $1 \le i \le k$. By construction, we have that $elev_{R_i}(u) = elev_{\overline{R}}(u)$. Consider a neighbor v of u such that (u, v) is a steepest-descent edge incident on u in R_i, and $level(R_i, v)$ is minimal among all such neighbors v of u. Since $elev_{\overline{R}}(v) \le$

$elev_{R_i}(v) \leq elev_{R_i}(u) = elev_{\overline{R}}(u)$ and $level(R_i, v) = level(R_i, u) - 1$, it holds that v has smaller symbolic elevation than u. Therefore, by induction, $v \in \mathcal{W}_{\overline{R}}(Q)$. If v is still a steepest-descent neighbor of u in $\overline{R}$, then this implies $u \in \mathcal{W}_{\overline{R}}(Q)$. Otherwise, there is a node $\widehat{v}$ such that $\sigma_{\overline{R}}(u, \widehat{v}) > \sigma_{\overline{R}}(u, v) \geq 0$. There must be an R_j such that $\widehat{v} \in \mathcal{W}_{R_j}(q_j)$, since otherwise, by construction of the watershed-overlay, we have $elev_{\overline{R}}(\widehat{v}) = high(\widehat{v}) \geq elev_{R_i}(\widehat{v})$ and thus, $\sigma_{R_i}(u, \widehat{v}) \geq \sigma_{\overline{R}}(u, \widehat{v}) > \sigma_{\overline{R}}(u, v) \geq \sigma_{R_i}(u, v)$ and v would not be a steepest-descent neighbor of u in R_i. Moreover, we have $\sigma_{\overline{R}}(u, \widehat{v}) > 0$ and, therefore, $elev_{\overline{R}}(\widehat{v}) < elev_{\overline{R}}(u)$, so $\widehat{v}$ has smaller symbolic elevation than u. Therefore, by induction, also $\widehat{v} \in \mathcal{W}_{\overline{R}}(Q)$ and thus, $u \in \mathcal{W}_{\overline{R}}(Q)$. □

The above lemma implies that for any set of nodes Q, the watershed-overlay $\overline{R}$ of the watersheds of the elements of Q in all possible realizations $\mathcal{R}_T$, would realize the potential watershed of Q. That is, we have that $\mathcal{W}_{\cup}(Q) \subseteq \mathcal{W}_{\overline{R}}(Q)$ and since $\mathcal{W}_{\cup}(Q)$ is the union of all watersheds of Q in all realizations, we also have that $\mathcal{W}_{\overline{R}}(Q) \subseteq \mathcal{W}_{\cup}(Q)$, which implies the equality of the two sets. Therefore, we call $\overline{R}$ the ***canonical realization*** of the potential watershed $\mathcal{W}_{\cup}(Q)$ and we denote it with $R_{\cup}(Q)$.

Note, however, that it is not immediately clear that the canonical realization always exists: the set of possible realizations is a non-discrete set, and thus the elevations in the canonical realization are defined as minima over a non-discrete set. Therefore, one may wonder if these minima always exist. Below, we will describe an algorithm that can actually compute the canonical realization of any set of nodes Q; from this we may conclude that it always exists.

4.1.2 Outline of the potential watershed algorithm

Next, we describe how to compute $\mathcal{W}_{\cup}(Q)$ and its canonical realization $R_{\cup}(Q)$ for a given set of nodes Q. Note that for all nodes $p \notin \mathcal{W}_{\cup}(Q)$, we have, by definition of the canonical realization, $elev_{R_{\cup}(Q)}(p) = high(p)$. The challenge is therefore to compute $\mathcal{W}_{\cup}(Q)$ and the elevations of the nodes of $\mathcal{W}_{\cup}(Q)$. Below we describe an algorithm that does this.

The idea of the algorithm is to compute the nodes of $\mathcal{W}_{\cup}(Q)$ and their elevations in the canonical realization in increasing order of elevation, similar to the way in which Dijkstra's shortest path algorithm computes distances from the source. The complete algorithm is laid out in Algorithm 1. The correctness and running time of the algorithm are proved in Theorem 2. A key ingredient of the algorithm is a subroutine, $\textsc{Expand}(q', z')$, which is defined as follows.

Definition 2. *Let* $\textsc{Expand}(q', z')$ *denote a function that returns for a node* q' *and an elevation* $z' \in [low(q'), high(q')]$ *a set of pairs of nodes and elevations, which includes the pair* (p, z) *if and only if* $p \in N(q')$*, there is a realization* R *with* $elev_R(q') \in [z', high(q')]$ *such that* $p \xrightarrow{\overline{R}} q'$*, and* z *is the minimum elevation of* p *over all such realizations* R*.*

Algorithm 1 PotentialWS(Q)

```
for all q ∈ Q do enqueue (q, z) with key z = low(q)
while the queue is not empty do
    Extract a pair (q', z') with minimum key z' from the queue
    if q' is not already in the output set then
        Output q' with elevation z'
        Enqueue each (p, z) ∈ Expand(q', z')
    end if
end while
```

4.1.3 Expansion of a node using the slope diagram

Before presenting the algorithm for the expansion of a node, we discuss a data structure that allows us to do this efficiently.

Definition 3. *For given elevations of the neighbors of a node p, we define the* **slope diagram** *of p as the set of points $\widehat{q}_i = (\delta_i, z_i)$ such that q_i is a neighbor of p, z_i is its elevation and δ_i is its distance to p.*

The intuition behind the slope diagram is the following. For a given elevation z of p, let $\widehat{p} = (0, z)$ be a point on the vertical axis of the slope diagram. Note that for any neighbor q_i, the slope of the line through p and q_i is the same as the slope of the line through $\widehat{p}$ and $\widehat{q}_i$ in the slope diagram. If q_i is a steepest descent neighbor of p under the given assignment of elevations, then all other neighbors $\widehat{q}_j$ lie above or on the line through $\widehat{p}$ and $\widehat{q}_i$ in the slope diagram.

Now, let $q_1, q_2, ...$ be a subset of the neighbors of p indexed such that $\widehat{q}_1, \widehat{q}_2, ...$ appear in counter-clockwise order along the boundary of the convex hull of the slope diagram, starting from the leftmost point and continuing to the lowest point. We ignore neighbors that do not lie on this lower left chain. Let H_i be the halfplane in the slope diagram that lies above the line through $\widehat{q}_i$ and $\widehat{q}_{i+1}$. Let $U(p)$ be the intersection of these halfplanes $H_1, H_2, ...$ with the halfplane to the right of the vertical line through the leftmost point, and the halfplane above the horizontal line through the bottommost point of the convex chain; see the shaded area in Figure 7.

The tangent of $U(p)$ through $\widehat{p}$ in the slope diagram passes through exactly the neighbors of p which are steepest descent neighbors of p. If $U(p)$ does not have a tangent through p, then p is a local minimum.

For a neighbor p of q', we can now compute the elevation of p as it should be returned by Expand(q', z') as follows. We use the slope diagram of p with the neighbors of p set to their highest position (that is, for a neighbor q_i we use $high(q_i)$) and compute the tangents to $U(p)$ which pass through the point $\widehat{q'} = (\delta', z')$, where δ' is the distance from q' to p. Assume for now that $U(p)$ has two tangents through $\widehat{q'}$ and let $[low', high']$ be the interval where the two tangents intersect the vertical axis of the slope diagram. Lemma 5 below implies that the lowest elevation that p can have in order to send flow via q' to the watershed is the lower endpoint of the interval $\mathcal{I} = [low(p), high(p)] \cap [low', high']$. This is the elevation

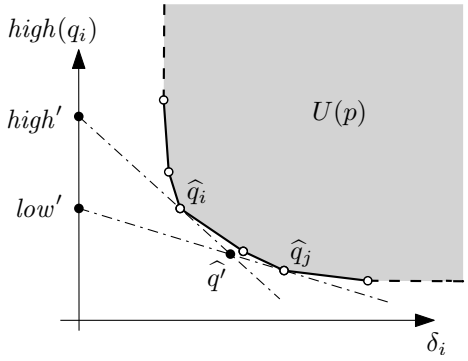


Figure 7: Querying the slope diagram.

which we return for p in the output of $\textsc{Expand}(q', z')$, unless the interval is empty. In the latter case we omit p from the output.

Lemma 5. *If q' is at elevation z', then the interval $\mathcal{I} = [low(p), high(p)] \cap [low', high']$ defines the elevations of p for which q' can be the steepest descent neighbor of p.*

Proof. Fix p at some arbitrary elevation z and let $\widehat{p} = (0, z)$ be its corresponding point in the slope diagram. If $z \in \mathcal{I}$, then all neighbors of p lie above or on the line through $\widehat{p}$ and $\widehat{q'}$ in the slope diagram. Thus, q' is steepest descent neighbor for this configuration of elevations. On the other hand, if there exists a configuration of the elevations of the other neighbors of p, such that they lie above or on the line through $\widehat{p}$ and $\widehat{q'}$, then they also lie above or on this line if we set them to their highest possible position. Thus, z would be included in $\mathcal{I}$ in this case. We conclude that if and only if $z \in [low', high']$ we can find a configuration of the elevations of the neighbors of p, in which q' is at elevation z' and at the same time q' is the steepest descent neighbor of p. □

We can compute the slope diagrams of all nodes with the neighbors set to their highest positions in a preprocessing phase. During the main algorithm the tangents can be computed via a binary search on the boundary of the convex hull in the slope diagram. In the proof of the following lemma we describe the technical details of this procedure more specifically. We also discuss the special cases where $U(p)$ does not have two tangents through $\widehat{q'}$.

Lemma 6. *Given the slope diagrams of the neighbors of q', we can compute the function $\textsc{Expand}(q', z')$ in time $O(d \log d')$, where d is the node degree of q', and d' is the maximum node degree of a neighbor of q'.*

Proof. For each neighbor p of q' we proceed as follows.

First, determine by binary search on the boundary of $U(p)$, if the vertical line through $\widehat{q'}$ intersects the boundary of $U(p)$, and if it does, whether the intersection point lies below $\widehat{q'}$. If it does, $\widehat{q'}$ lies in the interior of $U(p)$. Then q' can never be a steepest-descent neighbor of p, and therefore p is not included in the result of $\textsc{Expand}(q', z')$.

Otherwise $\widehat{q'}$ lies outside the interior of $U(p)$ and we continue as follows. If $\widehat{q'}$ lies on the vertical line that contains the left edge of $U(p)$, we define $high' = \infty$. Otherwise we find, by binary search on the boundary of $U(p)$, the corner $\widehat{q}_i$ that lies to left of the vertical line through $\widehat{q'}$, such that the line through $\widehat{q'}$ and $\widehat{q}_i$ is tangent on $U(p)$; let $high'$ be the vertical coordinate of the intersection of this tangent with the vertical axis. If $\widehat{q'}$ lies on or below the horizontal line that contains the bottom edge of $U(p)$, then we define $low' = z'$. Otherwise we find, by binary search on the boundary of $U(p)$, the corner $\widehat{q}_j$ that lies below the horizontal line through $\widehat{q'}$, such that the line through $\widehat{q'}$ and $\widehat{q}_j$ is tangent on $U(p)$; let low' be the vertical coordinate of the intersection of this tangent with the vertical axis.

If $low' > high(p)$ or $high' < low(p)$, then the set of elevations that p could have while having q' as a steepest-descent neighbor is empty, and we do not include p in the result of $\textsc{Expand}(q', z')$. Otherwise we include p with elevation $\max(low(p), low')$.

All computations for a single neighbor p of q' can be done in time logarithmic in the degree of p, and thus, the function $\textsc{Expand}(q', z')$ can be computed in time $O(d \log d')$ in total. □

4.1.4 Correctness and running time of the complete algorithm

Theorem 2. *After precomputations in $O(n \log n)$ time and $O(n)$ space, the algorithm* $\textsc{PotentialWS}(Q)$ *computes the potential watershed $\mathcal{W}_\cup(Q)$ of a set of nodes Q and its canonical realization $R_\cup(Q)$ in time $O(n \log n)$, where n is the number of edges in the terrain.*

Proof. The algorithm searches the graph starting from the nodes of Q. At each point in time we have three types of nodes. Nodes that have been extracted from the priority queue have a *finalized* elevation, a node that is currently in the priority queue but was never extracted (yet) has a *tentative* elevation, other nodes have not been reached.

We will show that when (p, z) is first extracted from the priority queue in Algorithm 1, p is indeed contained in the potential watershed of Q, and the elevation z is the lowest possible elevation of p such that water flows from p to any node in Q in any realization. To this end we use an induction on the points extracted, in the order in which they are extracted for the first time. The induction hypothesis consists of two parts:

(i) There exists a realization R and $q \in Q$ such that $elev_R(p) = z$, and R induces a flow path π from p to q which only visits nodes that have been extracted from the priority queue.

(ii) There exists no realization R and $q \in Q$ such that $elev_R(p) < z$ and $p \overrightarrow{R} q$.

If a node $p \in Q$ is extracted with $z = low(p)$, then the claims hold trivially. Note that the first extraction from the priority queue must be of this type.

If p is extracted from the priority queue for the first time and $p \notin Q$, then there must be at least one node p' that was extracted earlier, such that $\textsc{Expand}(p', z')$, for some elevation z', resulted in p having the tentative elevation z. By induction, there exists a realization R' and $q \in Q$, such that $elev_{R'}(p') = z'$, there is a flow path π from p' to q in R', and π does not include p.

To see that part (i) of the induction hypothesis holds for p, we construct a realization R by modifying R' as follows: we set $elev_R(p) = z$, and we set $elev_R(r) = high(r)$ for each neighbor r of p that does not lie on π. In comparison to R', only p and its neighbors may have a different elevation in R. Since $elev_R(p) = z \geq z'$ is still at least as high as the elevation of any node on π, water will still flow along the path π from p' to q. By the definition of Expand, none of the neighbors of p that are set at their highest elevation can out-compete p' as a steepest-descent neighbor of p. Therefore, in R, the node p must have p' or another node of π as a steepest-descent neighbor. Thus, water from p will flow onto π, and thus, to q.

Next we show (ii). Suppose, for the sake of contradiction, there is a realization R such that $elev_R(p) < z$ and there is a flow path from p to a node $q \in Q$. Consider two consecutive nodes r and s on this path, such that r has not been extracted before but s has been previously extracted (it may be that $r = p$ and/or $s \in Q$). Note that flow paths have to be monotone in the elevation. We argue that this path cannot stay below z in any realization. Since r is a neighbor of s, it has been added to the priority queue during the expansion of s. Let the tentative elevation of r that resulted from this expansion be z_r. By induction, since the elevation of s is finalized, z_r is a lower bound on the elevation of r for any flow path that follows the edge (r, s) and then continues to a node in Q in any realization. However, $z_r \geq z$, since r was not extracted from the priority queue before p. Therefore, a path from p to q that contains r with $elev_R(p) < z$ cannot exist. This proves (ii).

It follows that the algorithm outputs all nodes of $\mathcal{W}_\cup(Q)$ together with their elevations in $R_\cup(Q)$.

As for the running time, computing and storing $U(p)$ for a node p of degree d takes $O(d \log d)$ time and $O(d)$ space. Since the sum of all node degrees is $2n$, computing and storing $U(p)$ for all nodes p thus takes $O(n \log d_{\max})$ time and $O(n)$ space in total, where $d_{\max}$ is the maximum node degree in the terrain. While running algorithm PotentialWS(Q), each node is expanded at most once. By Lemma 6, Expand(q', z') on a node q' of degree d takes time $O(d \log d_{\max})$. Thus, again using that all nodes together have total degree $2n$, the total time spent on expanding is $O(n \log d_{\max}) = O(n \log n)$. Each extraction from the priority queue takes time $O(\log n)$ and there are at most $O(n)$ nodes to extract. Therefore PotentialWS takes time $O(n \log n)$ overall. □

For grid terrains, $d_{\max} = O(1)$, and thus, the slope diagram computations take only $O(1)$ time per expansion. In fact, since we only need to expand nodes that are in $\mathcal{W}_\cup(Q)$, we could actually compute $\mathcal{W}_\cup(Q)$ in $O(k \log k)$ time, where $k = |\mathcal{W}_\cup(Q)|$. Alternatively, we can use the techniques from Henzinger et al. [16] for shortest paths to overcome the priority queue bottleneck, and obtain the following result (details in Appendix A):

Theorem 3. *The canonical realization of the potential watershed of a set of cells Q in an imprecise grid terrain of n cells can be computed in $O(n)$ time.*

4.2 Potential downstream areas

Similar to the potential watershed of a set Q, we can define the set of points that potentially *receive* water from a node in Q. Let the ***potential downstream area*** of Q be defined as:

$$\mathcal{D}_\cup(Q) = \bigcup_{R\in\mathcal{R}_T} \bigcup_{q\in Q} \{p : q \xrightarrow[R]{} p\}.$$

Naturally, a canonical realization for this set does not necessarily exist. Nevertheless, the potential downstream area can be computed in a similar way as described in Section 4.1. The difference is that we will now process nodes in *decreasing* order of their *maximal* elevation such that they could still *receive* water from a node in Q. The algorithm is the same as Algorithm 1, except that in the first line the nodes are enqueued with their highest possible elevation, in line 3 we dequeue the current node with the largest key and we use the following subroutine in line 6:

Definition 4. *Let* ExpandDown(q', z') *denote a function that returns for a node* q' *and an elevation* $z' \in [low(q'), high(q')]$ *a set of pairs of nodes and elevations, which includes the pair* (p, z) *if and only if* $p \in N(q')$*, there is a realization* R *with* $elev_R(q') \in [low(q'), z']$ *such that* $q' \xrightarrow[R]{} p$*, and* z *is the maximum elevation of* p *over all such realizations* R*.*

Lemma 7. *We can compute the function* ExpandDown(q', z') *in* $O(d \log d)$ *time, where* d *is the node degree of* q'*.*

Proof. Consider the slope diagram of q' as defined in Section 4.1.3. Let z_0 be $\min high(p)$ over all neighbors p of q'; note that this is the vertical coordinate of the lowermost point of $U(q')$. Let $\widehat{q'} = (0, z')$ and consider its lower tangent to $U(q')$. Let $\widehat{p_i}$ be the corner of $U(q')$ that intersects the tangent. Similarly, let $\widehat{p_j}$ be the corner of $U(q')$ that intersects the tangent through $(0, \max(low(q'), z_0))$. Let $W(q')$ be the intersection of the halfplanes above these two tangents and the halfplanes $H_i, \ldots, H_j$ as defined in Section 4.1.3. Clearly, a neighbor of q' can have a steepest-descent edge from q', for some elevation of q' in $[low(q'), z']$, if and only if its representative in the slope diagram lies below $W(q')$ or on the boundary of $W(q')$. To compute the neighbors of q' and their elevations as they should be returned by ExpandDown(q', z'), we test each neighbor p of q' as follows. We find the point $\widehat{p'} = (|pq'|, z)$ that is the projection from $\widehat{p}$ down onto the boundary of $W(q')$. If $z \geq low(p)$, we return (p, z), otherwise we do not include p in the result.

The slope diagram with $W(q')$ can be computed $O(d \log d)$ time. The neighbors p of q' can be sorted by increasing distance from q' in the xy-projection in $O(d \log d)$ time; after that, the projections of all points $\widehat{p}$ can be computed in $O(d)$ time in total by handling them in order of increasing distance from q' and walking along the boundary of $W(q')$ simultaneously. □

Theorem 4. *Given a set of nodes* Q *of an imprecise terrain, we can compute the set* $\mathcal{D}_\cup(Q)$ *in time* $O(n \log n)$*, where* n *is the number of edges in the terrain.*

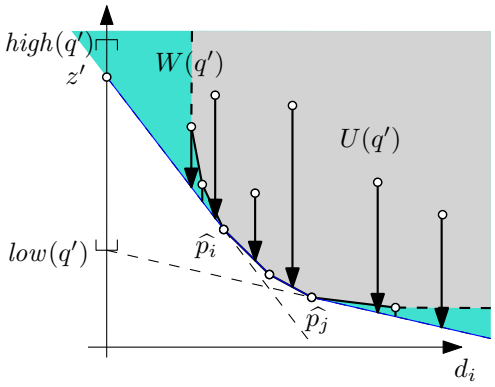


Figure 8: Computations in the slope diagram

Proof. The algorithm searches the graph starting from the nodes of Q. As in the algorithm for potential watersheds, nodes that have been extracted from the priority queue have a *finalized* elevation; nodes that are currently in the priority queue but were never extracted (yet) have *tentative* elevations. However, this time these elevations are not to be understood as elevations of the nodes in a single realization, but simply as the highest known elevations so that the nodes may be reached from Q.

The induction hypothesis is symmetric to the hypothesis used for potential watersheds: we show that when (p, z) is first extracted from the priority queue, p is indeed contained in the potential downstream area of Q, and the elevation z is the highest possible elevation of p such that water flows from any node in Q to p in any realization. Again, the induction is on the points extracted, in the order in which they are extracted for the first time. The induction hypothesis consists of two parts:

(i) There exists a realization R and $q \in Q$ such that $elev_R(p) = z$, there is a flow path π from q to p in R, and π only visits nodes that have been extracted from the priority queue.

(ii) There exists no realization R and $q \in Q$ such that $elev_R(p) > z$ and $q \xrightarrow{R} p$.

If a node $p \in Q$ is extracted with $z = high(p)$, then the claims hold trivially. Note that the first extraction from the priority queue must be of this type.

If p is extracted from the priority queue for the first time and $p \notin Q$, then there must be at least one node p' that was extracted earlier, such that ExpandDown(p', z'), for some elevation z', resulted in p having the tentative elevation z. By induction, there exists a realization R' and $q \in Q$, such that $elev_{R'}(p') = z'$, there is a flow path π from q to p' in R', and π does not include p.

So far the proof is basically symmetric to that of Theorem 2. However, to see (i), we need a different construction. Let $z'' \leq z'$ be an elevation such that water flows from p' to p in the realization R'' with $elev_{R''}(p') = z''$, $elev_{R''}(p) = z$, and $elev_{R''}(p'') = high(p'')$ for all other nodes p''. Note that z'' exists by definition of ExpandDown. We now construct a realization R by modifying R' as follows: we set $elev_R(p') = z''$, we set $elev_R(p) = z$, and we set $elev_R(r) = high(r)$ for each neighbor r of p' such that $r \neq p$ and r does not

lie on π. In comparison to R', only two nodes in R may have lower elevation, namely p and p'. Therefore, water will still flow along the path π from q until it either reaches p', or a node that now has p or p' as a new steepest-descent neighbor. Thus, in any case, there is a flow path from q to either p or p'. If the flow path reaches p', then, by definition of ExpandDown, none of the neighbors of p' that are set at their highest elevation can out-compete p as a steepest-descent neighbor of p'. Of course, the neighbors of p' that lie on π cannot out-compete p either, since these neighbors have elevation at least as high as p'. Therefore, p must be a steepest-descent neighbor of p' in R', and water from p' will flow to p. Thus, in any case, water from q will reach p in R' along a path that is a prefix of π, followed by an edge to p. This proves part (i) of the induction hypothesis.

The proof of part (ii) is completely analogous to the proof of Theorem 2.

It follows that the algorithm outputs all nodes of $\mathcal{D}_\cup(Q)$. The running time analysis is analogous to Theorem 2. □

4.3 Persistent watersheds

In this section we will give a definition of minimal watersheds, and explain how to compute them. Recall that the potential (maximal) watershed of a node set Q is defined as the set of nodes that have some flow path to a node in Q. We can write this as follows:

$$\mathcal{W}_\cup(Q) = \left\{p : \ \exists\ \pi \in \Pi(\mathcal{R}_T), \pi \ni p\ \exists\ q \in Q : p \underset{\pi}{\rightarrow} q\right\}.$$

An analogous definition that would be consistent with the intuitive idea of a minimal watershed would be:

$$\mathcal{W}_\cap(Q) = \left\{p : \ \forall\ \pi \in \Pi(\mathcal{R}_T), \pi \ni p\ \exists\ q \in Q : p \underset{\pi}{\rightarrow} q\right\}.$$

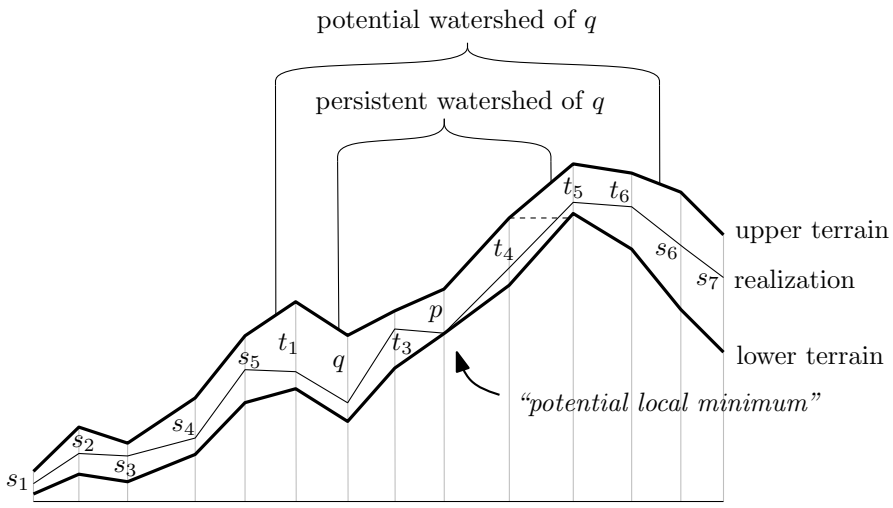
This is the set of nodes p from which water flows to Q via *any* induced maximal flow path that contains p. We call this the ***core watershed*** of Q.

However, this definition seems a bit too restrictive. Consider the case of a measuring device with a constant elevation error, used to sample points in a gently descending valley. It is possible that, by increasing the density of measurement points, we can create a region in which imprecision intervals of neighboring nodes overlap in the vertical dimension, and thus each node could become a local minimum in some realization. Thus, water flowing down the valley could, theoretically, "get stuck" at any point, and thus, the minimal watershed of any point q in the valley would contain nothing but q itself, see Box 4.1. [1]

Nevertheless, it seems clear that any water flowing in the valley must eventually reach q (possibly after flooding some local minima in the valley), since the water has nowhere else to go. This leads to an alternative definition of a minimal watershed, after we rewrite the definition of the core watershed slightly. Observe that the following holds for the

[1] Interestingly, there are some parallels to observations made in the GIS literature. Firstly, Hebeler et al. [15] observe that the watershed is more sensitive to elevation error in "flatlands". Secondly, simulations have shown that also potential local minima or "small sub-basins" can severely affect the outcome of hydrological computations [19].

Box 4.1 The persistent and potential watersheds of a node q.



An example of a 1.5 dimensional imprecise terrain, where the minimal watershed of q can be arbitrarily reduced by oversampling. In fact, the minimal watershed of q only contains q. Flow from any other node can get stuck in a potential local minimum. An example is the node p. Note that p cannot be in the minimal watershed of any other node. The complement of $\mathcal{W}_{\cup}(q)$ is the set $S = \{s_1, ..., s_7\}$. The q-avoiding potential watershed $\mathcal{W}_{\cup}^{\setminus q}(S)$ contains t_1 (because water from t_1 may flow directly to s_5) and t_5 and t_6 (because water from t_5 and t_6 may flow to s_6). The points q, t_3, p, t_4, t_5 are not in $\mathcal{W}_{\cup}^{\setminus q}(S)$, as water from there can only reach S by first flowing to q before reaching s_5. Thus, $\{q, t_3, p, t_4, t_5\}$ constitutes the persistent watershed $\mathcal{W}_{\cap}(q)$.

complement of the core watershed.

$$(\mathcal{W}_{\cap}(Q))^c = \left\{p : \ \exists\, \pi \in \Pi(\mathcal{R}_T), \pi \ni p \ \neg\exists\, q \in Q : p \underset{\pi}{\rightarrow} q\right\}$$

Thus, the core watershed of Q is the complement of the set of nodes p, for which it is possible that water follows a flow path from p that does *not* lead to Q. Recall that $\Pi^*(\mathcal{R}_T)$ is the set of all, not necessarily maximal, flow paths over all realizations in $\mathcal{R}_T$. Assume there exists a suitable set of *alternative destinations* S, such that we can rewrite the above equation as follows:

$$(\mathcal{W}_{\cap}(Q))^c = \left\{p : \ \exists\, \pi \in \Pi^*(\mathcal{R}_T), \pi \ni p\ \exists\, s \in S : (p \underset{\pi}{\rightarrow} s) \wedge (\pi \cap Q = \emptyset)\right\}.$$

Note that the right hand side is equivalent to the set:

$$\mathcal{W}_{\cup}^{\setminus Q}(S) := \bigcup_{\pi \in \Pi^*(\mathcal{R}_T)} \bigcup_{s \in S} \{p : (p \underset{\pi}{\rightarrow} s) \wedge (\pi \cap Q = \emptyset)\} \tag{1}$$

We call the set in Equation 1 the ***Q-avoiding potential watershed*** of a set of nodes S and we denote it with $\mathcal{W}_{\cup}^{\setminus Q}(S)$. This is the set of nodes that have a potential flow path to a node $s \in S$ that does not pass through a node of Q before reaching s.

It remains to identify the set of alternative destinations S. Since each flow path can be extended until it reaches a local minimum, the set of potential local minima clearly serves as such a set of destinations. Let $V_{\min}^{\setminus Q}$ be the union of all sets $P \subseteq V$ such that there exists a realization in which all nodes of P have the same elevation, P is a local minimum, and $P \cap Q = \emptyset$. However, it is also safe to include the nodes that do not have any flow path to Q, which is the complement of the set $\mathcal{W}_\cup(Q)$. It follows for the core watershed:

$$\mathcal{W}_\cap(Q) = \Big(\mathcal{W}_\cup^{\setminus Q}(\, V_{\min}^{\setminus Q} \cup (\, \mathcal{W}_\cup(Q))^c \,) \Big)^c$$

Note that we can rewrite this as follows:

$$\mathcal{W}_\cap(Q) = \Big(\mathcal{W}_\cup^{\setminus Q}(\, (\, \mathcal{W}_\cup(Q))^c \,) \Big)^c \setminus \mathcal{W}_\cup^{\setminus Q}(\, V_{\min}^{\setminus Q} \cap \mathcal{W}_\cup(Q) \,)$$

Based on the above considerations we suggest the following alternative definition of a minimal watershed.

Definition 5. *The* **persistent watershed** *of a set of nodes Q is defined as*

$$\mathcal{W}_{\text{⋒}}(Q) := \Big(\mathcal{W}_\cup^{\setminus Q}(\, (\, \mathcal{W}_\cup(Q))^c \,) \Big)^c.$$

This is the complement of the set of nodes that have a potential flow path to a node outside the potential watershed of Q without first passing through Q. An example can be seen in Box 4.1: the persistent watershed of q consists of the nodes that can never be high enough so that water from those nodes could escape from the potential watershed of q on the right; water from these nodes can only escape from the potential watershed of q by first flowing down to q.

To compute the persistent watershed efficiently, all we need are efficient algorithms to compute potential watersheds and Q-avoiding potential watersheds. We have already seen how to compute $\mathcal{W}_\cup(Q)$ efficiently in Section 4.1. Note that the Q-avoiding potential watershed of S is different from the potential watershed of S in the terrain T' that is obtained by removing the nodes Q and their incident edges from T. The next lemma states that we can also compute Q-avoiding potential watersheds efficiently.

Lemma 8. *There is an algorithm which outputs the Q-avoiding potential watershed of S and takes time $O(n \log n)$, where n is the number of edges of the terrain.*

Proof. We modify the algorithm to compute the potential watershed of S as shown in Algorithm 1, such that, each time the algorithm extracts a node from the priority queue, this node is discarded if it is contained in Q. Instead, the algorithm continues with the next node from the priority queue. Clearly, this algorithm does not follow any potential flow paths that flow through Q. However, the nodes of Q are still being considered by the neighbors of its neighbors as a node they have to compete against for being the steepest-descent neighbor. It is easy to verify that the proof of Theorem 2 also holds for the computation of Q-avoiding potential watersheds. □

By applying Theorem 2 and Lemma 8, we obtain:

Theorem 5. *We can compute the persistent watershed $\mathcal{W}_{\cap}(Q)$ of Q in time $O(n \log n)$, where n is the number of edges of the terrain.*

5 Regular terrains

We extend the results on imprecise watersheds in the network model for a certain class of imprecise terrains, which we call "regular". We will first define this class and characterize it. To this end we will introduce the notion of imprecise minima (see Definition 6), which are the "stable" minima of an imprecise terrain, regular or non-regular. In Section 5.2 we will describe how to compute these minima and how to turn a non-regular terrain into a regular terrain. In the remaining sections, we discuss nesting properties and fuzzy boundaries of imprecise watersheds. Furthermore, we observe that regular terrains have a well-behaved ridge structure, which delineates the main watersheds.

The main focus of this section is on the extension of the results in Section 4. Some of the concepts introduced here could also be applied to the surface model, however, we confine our discussion to the network model.

5.1 Characterization of regular terrains

We first give a definition of a proper minimum in an imprecise terrain.

Definition 6. *A set of nodes S in an imprecise terrain T is an* **imprecise minimum** *if and only if in every realization of T, the set S contains a local minimum, and no proper subset of S has this property.*

Note that the local minima contained in S can vary from one realization to another. Now a regular imprecise terrain is defined as follows:

Definition 7. *An imprecise terrain T is a* **regular imprecise terrain** *if and only if every local minimum of the lowermost realization R^- of T is an imprecise minimum of T.*

Any imprecise minimum S on a regular terrain is a minimum in R^-. Indeed, assume S would not be a minimum on R^-. Then, by Definition 6, it would still contain a proper subset S' that is a minimum on R^-. By Definition 7, S' must be an imprecise minimum of T, but this contradicts Definition 6. Now, we observe:

Lemma 9. *Let S be an imprecise minimum on a regular terrain. Then each node $s \in S$ has the same elevation lower bound $low(s)$. Furthermore, for each non-empty subset $S' \subset S$ we have $\mathcal{W}_{\cup}(S') = \mathcal{W}_{\cup}(S)$ and $\mathcal{W}_{R^-}(S') = \mathcal{W}_{R^-}(S)$.*

Proof. For the sake of contradiction, suppose not all nodes of S have the same elevation lower bound. Then there would be a proper subset S' of S that is a local minimum of R^-, and thus, by definition of a regular terrain, S' would be an imprecise minimum which is at the same time a proper subset of S. But this would, by Definition 6, contradict that S is an imprecise minimum. Therefore, each node $s \in S$ has the same elevation lower bound.

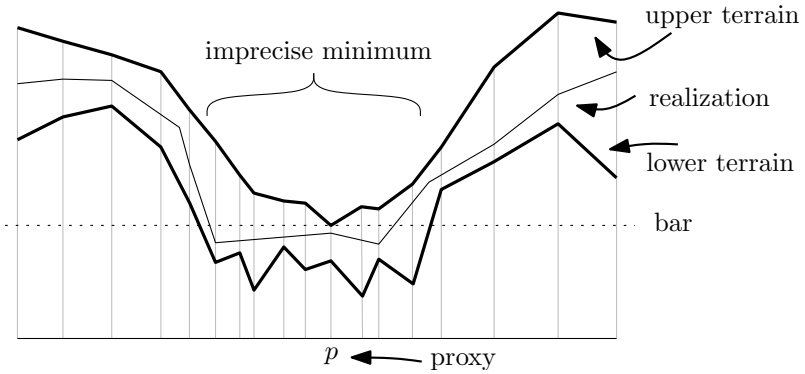


Figure 9: Example of an imprecise minimum with a proxy p in a non-regular terrain.

Now, in $R_\cup(S)$, all nodes $s \in S$ are at their lowermost elevation and thus S is a local minimum in $R_\cup(S)$. Thus, all nodes p that have a flow path to *any* node $s \in S$, have a flow path to *each* node $s \in S$, and thus each non-empty subset $S' \subset S$ has $\mathcal{W}_\cup(S') = \mathcal{W}_\cup(S)$. By the same argument, we have $\mathcal{W}_{R^-}(S') = \mathcal{W}_{R^-}(S)$. □

We will now derive a characterization of imprecise minima in general. For this, we introduce proxies.

Definition 8. *A* **proxy** *of an imprecise minimum S is a node $p \in S$, such that there are no realizations R and nodes $q \notin S$ such that $p \overrightarrow{R} q$.*

Thus, water that arrives in a proxy of an imprecise minimum S, can never leave S anymore. This implies that the proxy is not in the potential watershed of any set of nodes that lies entirely outside S. The following lemma states that every imprecise minimum contains a proxy.

Lemma 10. *Let the* **bar** *of a set S be $bar(S) = \min_{s\in S} high(s)$. A set S is an imprecise minimum if and only if (i) $bar(S) < \min_{t\in N(S)} low(t)$ and (ii) no proper subset S' of S has this property. Every imprecise minimum has a proxy.*

Proof. We first argue that if S is an imprecise minimum, then this implies (i) and (ii) for S.

To prove (i), consider the following realization R: For all nodes $r \in S$ we set $elev_R(r) = \max(bar(S), low(r))$, and for all nodes $t \in N(S)$ we set $elev_R(t) = low(t)$. Assume for the sake of contradiction that (i) would not hold. Then there exists a node $t \in N(S)$ which lies at elevation at most $bar(S)$ in R. Now, if all nodes of S would have the same elevation in R, then S would either be part of a local minimum that includes t, or S would have t as a lower neighbor: in either case, in the realization R the set S would neither be nor include a local minimum by itself, contradicting the assumption that S is an imprecise minimum. Therefore, since S is an imprecise minimum, it must be that not all nodes of S have the same elevation, and there exists a proper subset $S' \subset S$ which is a local minimum in R. Like all nodes of S, the local minimum S' must have elevation

at least $bar(S)$; each node $t \in N(S')$ must be set at a higher elevation $low(t)$. If we would remove the nodes of $N(S')$ from S, the imprecise minimum S would be separated into several components, including at least one component S'' that contains a node s with $high(s) = bar(S)$. This component S'' is a proper subset of S. Its neighborhood $N(S'')$ consists of nodes from $N(S)$ and $N(S')$, all of which have an elevation lower bound strictly above $bar(S) = \min_{s \in S''} high(s)$. Thus $S'' \subset S$ must contain a local minimum in any realization, contradicting the assumption that S is an imprecise minimum. Therefore the assumption that (i) would not hold must be wrong, and (i) must hold.

To prove (ii), assume, for the sake of contradiction, that S contains a proper subset S' such that $bar(S') < \min_{t \in N(S')} low(t)$. Thus, S' would contain a local minimum in any realization, and S would not be an imprecise minimum; hence (ii) must hold for S.

Now we argue that, if (i) and (ii) are met, then S is an imprecise minimum. Observe that condition (i) implies that S contains a local minimum in any realization. Now assume, for the sake of contradiction, that there exists a proper subset S' that always contains a local minimum. Let S' be a smallest such subset of S. We have that S' is an imprecise minimum, and therefore, as we proved above, it holds that $bar(S') < \min_{t \in N(S')} low(t)$, which contradicts that condition (ii) holds for S. Hence, there is no proper subset S' of S that always contains a local minimum; therefore S is an imprecise minimum.

As a proxy of an imprecise minimum S, we take any node s such that $high(s) < \min_{t \in N(S)} low(t)$. By condition (i) of the lemma, such a node s always exists. Since s lies below any node of $N(S)$ in any realization, there are no realizations R and nodes $q \notin S$ such that $s \xrightarrow{R} q$; thus s is a proxy of S. □

5.2 Computing proxies and regular terrains

Any imprecise terrain can be turned into a regular imprecise terrain by raising the lower bounds on the elevations such that local minima that violate the regularity condition are removed from R^-. Indeed, in hydrological applications it is common practice to preprocess terrains by removing local minima before doing flow computations [26]. To do so while still respecting the given upper bounds on the elevations, we can make use of the algorithm from Gray et al. [12]. The original goal of this algorithm is to compute a realization of a surface model that minimizes the number of local minima in the realization, but the algorithm can also be applied to a network model. It can easily be modified to output a proxy for each imprecise minimum of a terrain. Moreover, the realization M computed by the algorithm has the following convenient property: if we change the imprecise terrain by setting $low(v)$ to $elev_M(v)$ for each node, we obtain a regular imprecise terrain.

The algorithm The algorithm proceeds as follows. We will sweep a horizontal plane upwards. During the sweep, any node is in one of three states. Initially, each node is *undiscovered*. Once the sweep plane reaches $low(v)$, the state of the node changes to *pending*. Pending nodes are considered to be at the level of the sweep plane, but they may still be raised further. During the sweep, we will always maintain the connected components of the graph induced by the nodes that are currently pending; we call this graph G_P. As soon as

it becomes clear that a node cannot be raised further or does not need to be raised further, its final elevation on or below the sweep plane is decided and the node becomes *final*. More precisely, the algorithm is driven by two types of events: we may reach $low(v)$ for some node v, or we may reach $high(v)$ for some node v. These events are handled in order of increasing elevation; $low(v)$-events are handled before $high(v)$-events at the same elevation. The events are handled as follows:

- reaching $low(v)$: we make v pending, and find the component S of G_P that contains v. If v has a neighbor that is final, we make all nodes of S final at elevation $low(v)$.
- reaching $high(v)$: if v is final, nothing happens; otherwise we report v as a proxy, we find the connected component S of G_P that contains v, and we make all nodes of S final at elevation[2] $\max_{s\in S} low(s)$.

Gray et al. explain how to implement the algorithm to run in $O(n \log n)$ time [12].

Lemma 11. *Given an imprecise terrain T, (i) all nodes reported by the above algorithm are proxies of imprecise minima, and (ii) the algorithm reports exactly one proxy of each imprecise minimum of T.*

Proof. We first prove the second part, and then the first part of the lemma.

(ii) Let S be an imprecise minimum. Let v be the node in S which was the first to have its $high(v)$-event processed. By Lemma 10, v is a proxy of S and we have $high(v) < min_{t\in N(S)} low(t)$. Hence, when $high(v)$ is processed, the component of G_P that contains v does not contain any nodes outside S, and the $high(v)$-event is the first event to make any nodes in this component final. Thus, v is reported as a proxy. Furthermore, no node $s \in S$ can have $low(s) > high(v)$, otherwise $bar(S \setminus \{s\}) = high(v) < \min_{t\in\{s,N(S)\}} low(t) \leq \min_{t\in N(S\setminus\{s\})} low(t)$, and thus, by Lemma 10, S would not be an imprecise minimum. Hence, when the $high(v)$-event is about to be processed, all nodes of S have been discovered and are currently pending. The $high(v)$-event makes all nodes of S final; thus, any $high(s)$-events for other nodes $s \in S$ will remain without effect and no more proxies of S will be reported.

(i) Let v be a node that is reported as a proxy in a $high(v)$-event. We claim that the connected component S of G_P that contains v at that time, is an imprecise minimum. Indeed, by definition of G_P, all nodes of S are pending, and thus $high(v) = \min_{s\in S} high(s) = bar(S)$. Furthermore, because S is a connected component of G_P, all nodes $t \in N(S)$ must be either undiscovered or final. In fact, the algorithm maintains the invariant that no neighbor of a finalized node is pending; since all nodes in S are pending, all nodes $t \in N(S)$ must be undiscovered. Therefore $high(v) \leq \min_{t\in N(S)} low(t)$. Because all $low(t)$-events at the same elevation as $high(v)$ are processed before the $high(v)$-event is processed, we actually have a strict inequality: $high(v) < \min_{t\in N(S)} low(t)$. It follows that S satisfies condition (i) of Lemma 10. Furthermore, no proper subset S' of S has this property, otherwise, by the

[2]This is a small variation: the algorithm as described originally by Gray et al. would make the elevations final at $high(v) = \min_{s\in S} high(s)$. However, in the current context we prefer to make the elevations final at $\max_{s\in S} low(s)$, to maintain as much of the imprecision in the original imprecise terrain as possible.

analysis given above, a proxy for S' would have been reported already and the nodes from S' would have been removed from G_P at that time. Hence, S also satisfies condition (ii) of Lemma 10, and S is an imprecise minimum, with v as a proxy. □

Lemma 12. *Let M be the realization of a terrain T as computed by the algorithm described above. Let T' be the imprecise terrain that is obtained from T by setting $low(v) = elev_M(v)$ for each node v. The terrain T' is a regular imprecise terrain.*

Proof. Note that M is the lowermost realization of T'. Consider any local minimum S of M. Observe that the algorithm cannot have finalized the elevations of the last pending nodes of S in a $low(v)$-event, because then we would have $v \in S$ and v must have a neighbor $t \notin S$ that was finalized before v; hence $elev_M(t) \leq elev_M(v)$ and S would not be a local minimum. Therefore, the algorithm must have finalized the last elevations of the nodes of S in a $high(v)$-event for a node $v \in S$. Furthermore, each node $t \in N(S)$ must have been undiscovered at that time; otherwise t would have become part of the same component as the nodes of S before its elevations were finalized, or t would have been finalized before v: in both cases S would not be a local minimum. Hence we have $low(t) > high(v)$ for each node $t \in N(S)$, and thus, S is a local minimum in every realization of T or T'. Furthermore, no proper subset of S' of S contains a local minimum in every realization of T', since in particular, in M the set S is a local minimum and therefore no proper subset S' of S is a local minimum. Thus, by Definition 6 and Definition 7, T' is a regular terrain. □

5.3 Nesting properties of imprecise watersheds

To be able to design data structures that store imprecise watersheds and answer queries about the flow of water between nodes efficiently, it would be convenient if the watersheds satisfy the following ***nesting condition***: if p is contained in the watershed of q, then the watershed of p is contained in the watershed of q. Clearly, potential watersheds do not satisfy this nesting condition, while core watersheds do. However, in general, persistent watersheds are not nested in this way. We give a counter-example that uses a non-regular terrain in the next lemma, before proving the nesting condition for persistent watersheds in regular terrains later in this section.

Lemma 13. *There exists an imprecise terrain with two nodes p and q such that $p \in \mathcal{W}_\cap(q)$ and $\mathcal{W}_\cap(p) \not\subseteq \mathcal{W}_\cap(q)$.*

Proof. We give an example of a non-regular terrain that has this property. Refer to Figure 10. The persistent watershed of p as shown in red is not completely contained in the persistent watershed of q as shown in blue. The left figure gives a top-view. All edges have unit length, except for the edge between w and q. The right figure shows the fixed elevations of s, t, t', u, v and w, the elevation intervals of p, q and r, and the correct horizontal distances on all edges except $|pv|$ and $|qv|$. The red outline delimits $\mathcal{W}_\cup(p) = \{p, q, r, s, t, v, w\}$. The red dashed outline delimits $\mathcal{W}_\cap(p) = \{p, s, v\}$. The blue outline delimits $\mathcal{W}_\cup(q) = \{p, q, s, v, w\}$. The blue dashed outline delimits $\mathcal{W}_\cap(q) = \{p, q, v\}$. □

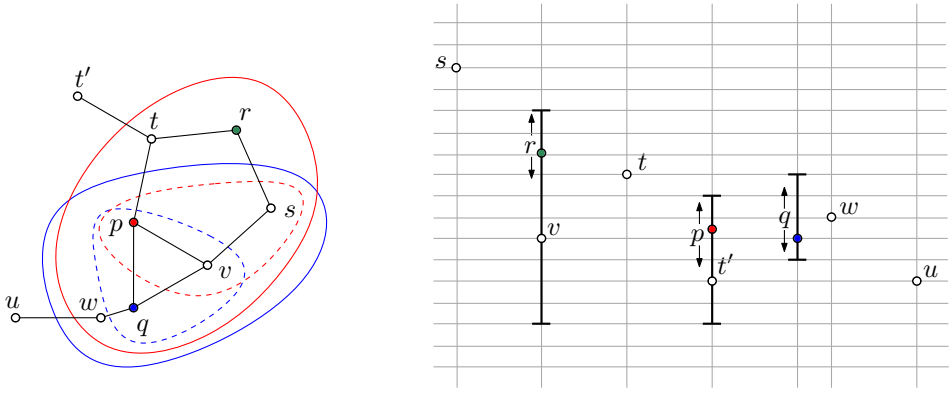


Figure 10: Counterexample of a non-regular terrain to the nesting condition of persistent watersheds.

The following lemmas will prove that on regular imprecise terrains persistent watersheds do satisfy the nesting condition.

Lemma 14. *Let Q be a set of nodes of a regular imprecise terrain, then $\mathcal{W}_{\cap}(Q) \subseteq \mathcal{W}_{R^-}(Q)$.*

Proof. Consider a maximal flow path π from a node $p \in \mathcal{W}_{\cap}(Q)$ in R^-. We claim that it flows to a node of Q. In particular, we claim that π cannot flow to a local minimum S in R^- without reaching any node of Q. We prove this claim by contradiction. Thus, assume π would be a Q-avoiding flow path to a local minimum S such that $Q \cap S = \emptyset$. Because the terrain is regular, S must be an imprecise minimum (by Definition 7), and therefore S must have a proxy by Lemma 10. Thus, π flows to a proxy $s \in S$. As observed above, s is not in the potential watershed of any set of nodes outside S; in particular, s is not in $\mathcal{W}_{\cup}(Q)$, see Figure 11 (left). This implies that in R^- there exists a flow path from p which leaves $\mathcal{W}_{\cup}(Q)$ without going through any node of Q. This contradicts the fact that $p \in \mathcal{W}_{\cap}(Q)$, since by the definition of persistent watersheds, π cannot leave $\mathcal{W}_{\cup}(Q)$ without going through Q.

Therefore, from any node $p \in \mathcal{W}_{\cap}(Q)$ there must be a flow path to a node $q \in Q$ in R^-, and thus, $\mathcal{W}_{\cap}(Q) \subseteq \mathcal{W}_{R^-}(Q)$. □

Lemma 15. *Let Q be a set of nodes of an imprecise terrain, and let $P \subseteq \mathcal{W}_{R^-}(Q)$. Then $\mathcal{W}_{\cup}(P) \subseteq \mathcal{W}_{\cup}(Q)$.*

Proof. Recall that $R_{\cup}(P)$ is the canonical realization of $\mathcal{W}_{\cup}(P)$ as defined in Section 4.1.1. Let $\overline{R}$ be the watershed-overlay of $\mathcal{W}_{R_{\cup}(P)}(P)$ and $\mathcal{W}_{R^-}(Q)$. Consider a node $r \in \mathcal{W}_{\cup}(P)$ and a flow path π from r to a node $p \in P$ in $R_{\cup}(P)$, see Figure 11 (center). Let π' be the maximal prefix of π such that the nodes of π' have the same elevation in $R_{\cup}(P)$ and $\overline{R}$, and let π'' be the maximal prefix of π' such that π'' is still a flow path in $\overline{R}$. We distinguish three cases:

- If both π' and π'' are empty, then r has higher elevation in $R_{\cup}(P)$ than in $\overline{R}$, so r must be in $\mathcal{W}_{R^-}(Q)$.

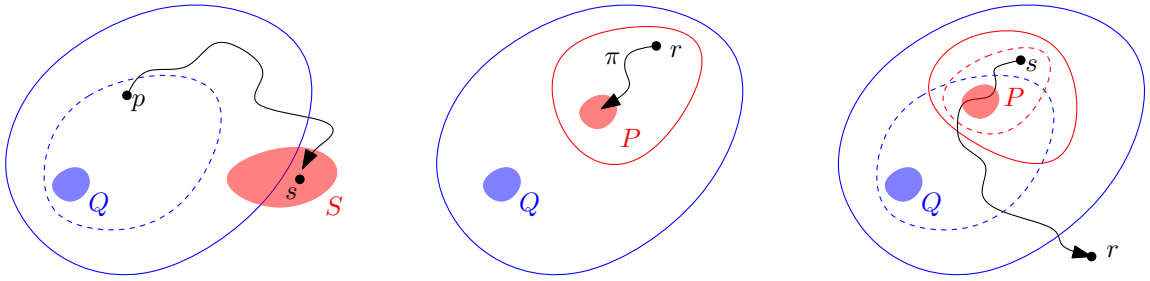


Figure 11: Illustrations to the proofs of Lemmas 14 (left), 15 (center) and 16 (right).

- If $\pi' = \pi'' = \pi$, then flow from r reaches a node $p \in P \subseteq \mathcal{W}_{R^-}(Q)$ in $\overline{R}$.
- Otherwise, let (u, v) be the edge of π such that u is the last node of π''. Now v is not on π'', so in $\overline{R}$, flow from u either still follows (u, v) but $elev_{\overline{R}}(v) < elev_{R_\cup(P)}(v)$, or flow from u is diverted over an edge $(u, \widehat{v})$ to another node $\widehat{v}$ with $elev_{\overline{R}}(\widehat{v}) < elev_{R_\cup(P)}(\widehat{v})$. In either case, from u we follow an edge to a node of which the elevation in $\overline{R}$ is lower than in $R_\cup(P)$; therefore this must be a node of $\mathcal{W}_{R^-}(Q)$.

In any case, there is a flow path from r to a node of $\mathcal{W}_{R^-}(Q)$. From here, there must be a path to a node $q \in Q$, since every flow path within $\mathcal{W}_{R^-}(Q)$ in R^- is also a flow path in $\overline{R}$. Thus there is a flow path from r to q in $\overline{R}$, and thus, $r \in \mathcal{W}_\cup(Q)$. This proves the lemma. □

Lemma 16. *(Persistent watersheds are nested) Let Q be a set of nodes of a regular imprecise terrain, and let $P \subseteq \mathcal{W}_\cap(Q)$. Then $\mathcal{W}_\cap(P) \subseteq \mathcal{W}_\cap(Q)$.*

Proof. Assume for the sake of contradiction that there exists a node $s \in \mathcal{W}_\cap(P)$, such that $s \notin \mathcal{W}_\cap(Q)$. Clearly, $s \in \mathcal{W}_\cup(P)$ and by Lemma 15 and Lemma 14, it holds that $s \in \mathcal{W}_\cup(Q)$. Furthermore, s must have a flow path π to a point $r \notin \mathcal{W}_\cup(Q)$, which does not pass through a node of Q, refer to Figure 11 (right). By Lemma 15 and Lemma 14, $\mathcal{W}_\cup(P) \subseteq \mathcal{W}_\cup(Q)$, and thus $r \notin \mathcal{W}_\cup(P)$. Furthermore, since $s \in \mathcal{W}_\cap(P)$, π must include a node $p \in P$. This contradicts the fact that $p \in \mathcal{W}_\cap(Q)$, since $\pi \cap Q = \emptyset$. □

5.4 Fuzzy watershed boundaries

Lemma 14 and Lemma 15 also allow us to compute the difference between the potential and the persistent watershed of a set of nodes Q efficiently, given only the boundary of the watershed of Q on the lowermost realization of the terrain. We first define these concepts more precisely.

Definition 9. *Given a realization R, and a set of nodes Q, let $\mathcal{X}_R(Q)$ be the set of directed edges (u, v) such that $u \in \mathcal{W}_R(Q)$ and $v \notin \mathcal{W}_R(Q)$. We call $\mathcal{X}_R(Q)$ the* **watershed boundary** *of Q in R. Likewise, we define the* **fuzzy watershed boundary** *of Q as the directed set of edges (u, v) such that $u \in \mathcal{W}_\cup(Q)$ and $v \notin \mathcal{W}_\cap(Q)$ and we denote it with $\mathcal{X}_\cup(Q)$. We call the set $\mathcal{W}_\cup(Q) \setminus \mathcal{W}_\cap(Q)$ the* **uncertainty area** *of this boundary.*

We will now discuss how we can compute the uncertainty area of any fuzzy watershed boundary efficiently.

Algorithm to compute the uncertainty area of a watershed. Assume we are given $\mathcal{X}_{R^-}(Q)$. We will compute the set $\mathcal{W}_\cup(Q) \setminus \mathcal{W}_\cap(Q)$ with Algorithm 1, modified as follows. Instead of initializing the priority queue with the nodes of Q, we initialize in the following way. For each edge $(u, v) \in \mathcal{X}_{R^-}(Q)$, we use the slope diagram of u to determine the minimum elevation z_u of u, such that there is a realization in which water flows on the edge from u to v. If there exists such an elevation z_u, we enqueue u with elevation (and key) z_u. Similarly, we use the slope diagram of v to determine the minimum elevation z_v of v, such that water may flow on the edge from v to u. If z_v exists, we enqueue v with elevation (and key) z_v. After initializing the priority queue in this way, we run Algorithm 1 as written.

Lemma 17. *If the terrain is regular, the algorithm described above computes $\mathcal{W}_\cup(Q) \setminus \mathcal{W}_\cap(Q)$. This is the uncertainty area of the fuzzy watershed boundary of Q.*

Proof. Observe, following the proof of Theorem 2, that for any node p output by the above algorithm there are a realization R_s and a node s which was in the initial queue with elevation z_s, such that $elev_{R_s}(s) = z_s$ and R_s induces a flow path π from p to s. Let (u, v) be the edge of $\mathcal{X}_{R^-}(Q)$ which led to the insertion of $s \in \{u, v\}$ into Q with elevation z_s. Let t be the other node of (u, v), that is, $t \in \{u, v\} \setminus \{s\}$. Let R_t be the realization obtained from R_s by setting $elev_{R_t}(t) = low(t)$. Observe that, by our choice of z_s, the realization R_t now induces a flow path from p to t. We will now argue that (i) $p \in \mathcal{W}_\cup(Q)$, and (ii) $p \notin \mathcal{W}_\cap(Q)$.

(i) The existence of R_u implies that $p \in \mathcal{W}_\cup(u)$; since $u \in \mathcal{W}_{R^-}(Q)$ (by definition of $\mathcal{X}_{R^-}(Q)$) this implies $p \in \mathcal{W}_\cup(Q)$ (by Lemma 15).

(ii) By definition of $\mathcal{X}_{R^-}(Q)$, there is no flow path from v to Q on R^-. Hence, any maximal flow path from v on R^- must lead to a local minimum S that does not contain any node of Q, and by Definition 7, each such local minimum S is an imprecise minimum. Now, by Lemma 10, each such local minimum S contains a proxy s, which is, by Definition 8, not contained in $\mathcal{W}_\cup(Q)$. Thus there is a flow path from v that does not go through any nodes of Q and leads to a proxy $s \notin \mathcal{W}_\cup(Q)$. Hence, by Definition 5, $p \notin \mathcal{W}_\cap(Q)$.

Next, we will argue that if $p \in \mathcal{W}_\cup(Q)$ and $p \notin \mathcal{W}_\cap(Q)$, the algorithm will output p. We distinguish two cases.

If $p \in \mathcal{W}_{R^-}(Q)$, then, because $p \notin \mathcal{W}_\cap(Q)$, there must be a flow path on R^- from p to a minimum S that does not contain any node of Q. By Definition 7, Lemma 10 and Definition 8, there will then be a flow path from p to a proxy $s \in S$ that lies outside $\mathcal{W}_\cup(Q)$, and thus, outside $\mathcal{W}_{R^-}(Q)$.

If $p \notin \mathcal{W}_{R^-}(Q)$, then, because $p \in \mathcal{W}_\cup(Q)$, there must be a realization in which there is a flow path from p to Q, and thus, from p to $\mathcal{W}_{R^-}(Q)$.

In both cases, there is a realization in which there is a flow path from p that traverses an edge $(u, v) \in \mathcal{X}_{R^-}(Q)$, either from u to v or from v to u. The algorithm reports at least all such points p.

This completes the proof of the lemma. □

Note that if k is the size of the input ($\mathcal{X}_{R^-}(Q)$) and the output ($\mathcal{W}_\cup(Q) \setminus \mathcal{W}_\cap(Q)$) and d' is the maximum node degree, then the above algorithm runs in $O(k \log k + k \log d')$ time. When a data structure is given that stores the boundaries of watersheds on R^- so that they can be retrieved efficiently, and the imprecision is not too high, this would enable us to compute the boundaries and sizes of potential and persistent watersheds much faster than by computing them (or their complements) node by node with Algorithm 1.

We can use the same idea as above to compute an uncertain area of the watershed boundaries between a set of nodes Q. More precisely, given a collection of nodes Q such that no node $q \in Q$ is contained in the potential watershed of another node $q' \in Q$, we can compute the nodes that are in the potential watersheds of multiple nodes from Q.

Algorithm to compute the uncertainty area between watersheds. Let Q be $\{q_1, ..., q_k\}$ and let G' be the graph induced by the potential watershed of Q. The algorithm is essentially the same as the algorithm that computes the uncertainty area of a single watershed's boundary—the main difference is that now we have to start it with a suitable set of edges $\mathcal{X}$ on the fuzzy boundaries *between* the watersheds of the nodes of Q. More precisely, $\mathcal{X}$ should be an edge separator set of G', which separates the nodes of G' into k components $G'_1, ..., G'_k$ such that nodes of each component G'_i are completely contained in $\mathcal{W}_\cup(q_i)$.

We obtain $\mathcal{X}$ with the following modification of Algorithm 1. For each node p we will maintain, in addition to a tentative elevation z, a tentative tag that identifies a node $q \in Q$ such that there is a realization R with $elev_R(p) = z$ and $p \overset{}{\underset{R}{\rightarrow}} q$. We initialize the priority queue of Algorithm 1 with all nodes $q \in Q$, each with tentative elevation $low(q)$ and each tagged with itself. The first time any particular node q' is extracted from the priority queue, we obtain not only its final elevation but also its final tag q from the queue, and each pair $(p, z) \in \textsc{Expand}(q', z')$ is enqueued with that same tag q. At the end of Algorithm 1, we obtain the set of nodes in $\mathcal{W}_\cup(Q)$ together with their elevations in the canonical realization $R_\cup(Q)$ and with tags, such that any set of nodes tagged with the same tag $q \in Q$ forms a connected subset of $\mathcal{W}_\cup(q)$. We now extract the separator set $\mathcal{X}$ by identifying the edges between nodes of different tags.

Having obtained $\mathcal{X}$, we compute the union of the pairwise intersections of the potential watersheds of $q_1, ..., q_k$ as follows. Again, we use Algorithm 1. This time the priority queue is initialized as follows. For each edge $(u, v) \in \mathcal{X}$, we use the slope diagram of u to determine the minimum elevation z_u of u, such that there is a realization R *with* $elev_R(v) = elev_{R_\cup(Q)}(v)$ in which water flows on the edge from u to v. If there exists such an elevation z_u, we enqueue u with elevation (and key) z_u. Similarly, we use the slope diagram of v to determine the minimum elevation z_v of v, such that water may flow on the edge from v to u *at elevation* $elev_{R_\cup(Q)}(u)$. If z_v exists, we enqueue v with elevation (and key) z_v. After initializing the priority queue in this way, we run Algorithm 1 as written, and output the result.

Lemma 18. *Given a set of nodes $q_1, \ldots, q_k$ of an imprecise terrain, such that $q_i \notin \mathcal{W}_\cup(q_j)$ for any $i \neq j$ and $1 \leq i, j \leq k$, we can compute the set $\bigcup_i \bigcup_{j \neq i} (\mathcal{W}_\cup(q_i) \cap \mathcal{W}_\cup(q_j))$ in*

$O(n \log n)$ time, where n is the number of edges of the imprecise terrain.

Proof. The separator set $\mathcal{X}$ is obtained in $O(n \log n)$ time by running the modified version of Algorithm 1 and one scan over the graph to identify edges between nodes with different tags. Computing the union of the pairwise intersections of the potential watersheds of $q_1, ..., q_k$ with the modified Algorithm 1 takes $O(n \log n)$ time again.

By the same arguments as in the proof of Lemma 17, we can observe the following: for any node p output by the above algorithm, there are an edge $(u, v) \in \mathcal{X}$, a realization R_u with $elev_{R_u}(u) = elev_{R_\cup(Q)}(u)$ and $p \overrightarrow{R_u} u$, and a realization R_v with $elev_{R_v}(v) = elev_{R_\cup(Q)}(v)$ and $p \overrightarrow{R_v} v$. Let $q_u, q_v \in Q$ be the nodes of Q with which u and v were tagged, respectively. It follows that there is a flow path from p to q_u in the watershed overlay of $\mathcal{W}_{R_u}(u)$ and $\mathcal{W}_{R_\cup(Q)}(q_u)$, so $p \in \mathcal{W}_\cup(q_u)$. Analogously, $p \in \mathcal{W}_\cup(q_v)$. Since $(u, v) \in \mathcal{X}$, we have $q_u \neq q_v$, so any point p that is output by the algorithm lies in the intersection of the potential watersheds of two different nodes from Q.

Next, we will argue that if p lies in the intersection of the potential watersheds of two different nodes from Q, then the algorithm will output p. Let $q \in Q$ be the node with which p is tagged (hence, $p \in \mathcal{W}_\cup(q)$), and let $q' \in Q, q' \neq q$ be another node from Q such that $p \in \mathcal{W}_\cup(q')$. Consider a flow path π from p to q' in $R_\cup(q')$, and let (r, r') be the edge on π such that r is tagged with a node other than q' while r' and all nodes following the last occurrence of (r, r') in π are tagged with q'. Note that (r, r') must exist because all nodes of π lie in $\mathcal{W}_\cup(Q)$ and have received a tag, p is tagged with another node than q', and, since none of the nodes of Q lie in each other's potential watersheds, q' is tagged with itself. Therefore (r, r') exists, and $(r, r') \in \mathcal{X}$. Moreover, we have $elev_{R_\cup(Q)}(r') = elev_{R_\cup(q')}(r')$. Therefore r was put in the priority queue with the minimum elevation such that there is a realization R with $elev_R(r') = elev_{R_\cup(q')}(r')$ in which water flows on the edge from r to r'. By induction on the nodes of π from r back to p, it follows that p must eventually be extracted from the priority queue and output.

This completes the proof of the lemma. □

5.5 The fuzzy watershed decomposition

In this section we further characterize the structure of imprecise terrains by considering the ridge lines that delineate the "main" watersheds. In fact, the fuzzy watershed boundaries (Definition 9) of the imprecise minima (Definition 6) possess a well-behaved ridge structure if the terrain is regular. Consider the following definition of an "imprecise" ridge.

Definition 10. *Let $S_1, \ldots, S_k$ be the imprecise minima of an imprecise terrain. We call the union of the pairwise intersection of the potential watersheds of imprecise minima the* **fuzzy ridge** *of the terrain.*

Let S be an imprecise minimum of a regular imprecise terrain. The next lemma testifies that the persistent watershed of any proxy q of S is equal to the intersection of the persistent watersheds of all possible non-empty subsets of S. Therefore, we think of $\mathcal{W}_\cap(q)$ as the actual minimal watershed of S, or the minimum associated with S. By Lemma 9,

the potential watersheds of all non-empty subsets of S are equal. Consequently, we think of the fuzzy watershed boundary of q as the fuzzy watershed boundary of S.

Lemma 19. *Let S be an imprecise minimum on a regular terrain, and let x be any proxy of S. Then $\bigcap_{\emptyset \subset S' \subseteq S} \mathcal{W}_{\cap}(S') = \mathcal{W}_{\cap}(x)$.*

Proof. Let C denote $\bigcap_{\emptyset \subset S' \subseteq S} \mathcal{W}_{\cap}(S')$, the intersection of the persistent watersheds of all non-empty subsets of S. Consider the complement of this set:

$$(C)^c := \left(\bigcap_{\emptyset \subset S' \subseteq S} \mathcal{W}_{\cap}(S') \right)^c = \bigcup_{\emptyset \subset S' \subseteq S} (\mathcal{W}_{\cap}(S'))^c = \bigcup_{\emptyset \subset S' \subseteq S} \mathcal{W}_{\cup}^{\setminus S'}((\mathcal{W}_{\cup}(S'))^c).$$

By Lemma 9 we have $\mathcal{W}_{\cup}(S') = \mathcal{W}_{\cup}(x) = \mathcal{W}_{\cup}(S)$ for any non-empty set $S' \subseteq S$, so we have:

$$(C)^c = \bigcup_{\emptyset \subset S' \subseteq S} \mathcal{W}_{\cup}^{\setminus S'}((\mathcal{W}_{\cup}(S))^c).$$

Now, by Definition 8, it is impossible for water that reaches x to continue to flow to a node outside of $\mathcal{W}_{\cup}(S)$. Therefore, any flow path to a node outside $\mathcal{W}_{\cup}(S)$ that avoids a non-empty set $S' \subseteq S$, also avoids x, and we have:

$$\mathcal{W}_{\cup}^{\setminus S'}((\mathcal{W}_{\cup}(S))^c) \subseteq \mathcal{W}_{\cup}^{\setminus x}((\mathcal{W}_{\cup}(S))^c) = \mathcal{W}_{\cup}^{\setminus x}((\mathcal{W}_{\cup}(x))^c).$$

Thus we get:

$$(C)^c = \bigcup_{\emptyset \subset S' \subseteq S} \mathcal{W}_{\cup}^{\setminus S'}((\mathcal{W}_{\cup}(S))^c) = \mathcal{W}_{\cup}^{\setminus x}((\mathcal{W}_{\cup}(x))^c).$$

By the definition of persistent watersheds, we now have $C = \mathcal{W}_{\cap}(x)$, which completes the proof.

□

We can now further characterize the fuzzy ridge for regular terrains. The following lemma implies that on a regular terrain, the fuzzy ridge is equal to the union of the uncertainty areas of the fuzzy watershed boundaries of any representative set of proxies of the imprecise minima, see Corollary 1.

Lemma 20. *Let $S_1, \ldots, S_k$ be the imprecise minima of a regular imprecise terrain and let $q_1, \ldots, q_k$ be associated proxies. For any $1 \leq i \leq k$, we have that $\mathcal{W}_{\cap}(q_i) = \left(\bigcup_{j \neq i} \mathcal{W}_{\cup}(q_j) \right)^c$.*

Proof. Since q_i is a proxy, we have that,

$$(\mathcal{W}_{\cap}(q_i))^c = \mathcal{W}_{\cup}^{\setminus q_i}((\mathcal{W}_{\cup}(q_i))^c) = \mathcal{W}_{\cup}((\mathcal{W}_{\cup}(q_i))^c) = \bigcup_{p \in (\mathcal{W}_{\cup}(q_i))^c} \mathcal{W}_{\cup}(p).$$

Now, for a node $p \in (\mathcal{W}_{\cup}(q_i))^c$, consider a minimum S that is reached by a flow path from p in R^-. By Definition 7, we have that S is an imprecise minimum, and since

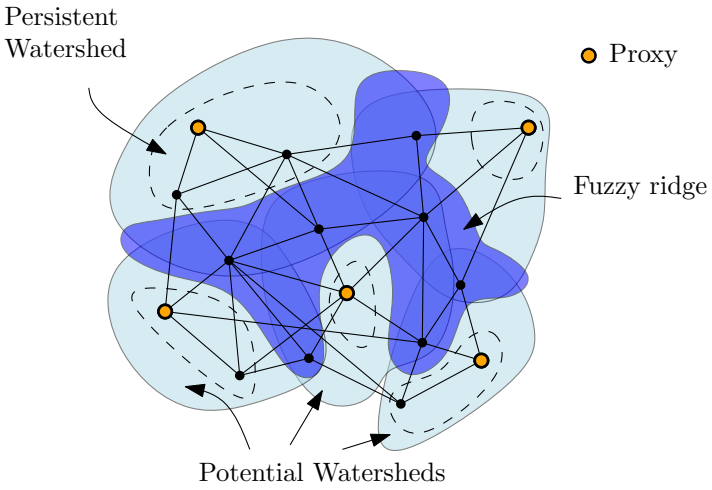


Figure 12: Illustration to the fuzzy ridge on a regular terrain.

$p \in (\mathcal{W}_\cup(q_i))^c = (\mathcal{W}_\cup(S_i))^c$, we have $S \neq S_i$. As such, S must be equal to some S_j for $j \neq i$. Furthermore, by Lemma 9 we have $\mathcal{W}_{R^-}(S_j) = \mathcal{W}_{R^-}(q_j)$, therefore $p \in \mathcal{W}_{R^-}(q_j)$. Now, Lemma 15 implies that $\mathcal{W}_\cup(p) \subseteq \mathcal{W}_\cup(q_j)$. It follows that $(\mathcal{W}_\cap(q_i))^c \subseteq \bigcup_{i\neq j} \mathcal{W}_\cup(q_j)$.

Since we also have that $q_j \in (\mathcal{W}_\cup(q_i))^c$ for any $j \neq i$, we also get $(\mathcal{W}_\cap(q_i))^c \supseteq \bigcup_{i\neq j} \mathcal{W}_\cup(q_j)$, which implies the equality. □

Corollary 1. *Lemma 20 implies that, given $q_1, \ldots, q_k$, a representative set of proxies for the imprecise minima of a regular imprecise terrain, it holds that*

$$\begin{aligned}\bigcup_i (\mathcal{W}_\cup(q_i) \setminus \mathcal{W}_\cap(q_i)) &= \bigcup_i \left(\mathcal{W}_\cup(q_i) \setminus \left(\bigcup_{j\neq i} \mathcal{W}_\cup(q_j) \right)^c \right) \\ &= \bigcup_i \left(\mathcal{W}_\cup(q_i) \cap \bigcup_{j\neq i} \mathcal{W}_\cup(q_j) \right) \\ &= \bigcup_i \bigcup_{j\neq i} (\mathcal{W}_\cup(q_i) \cap \mathcal{W}_\cup(q_j)) \,.\end{aligned}$$

By Lemma 9, this is equal to the fuzzy ridge of this terrain as defined in Definition 10. This relationship is illustrated in Figure 12.

Combining this with Lemma 11 and Lemma 18 we obtain:

Theorem 6. *We can compute the fuzzy ridge of a regular imprecise terrain in $O(n \log n)$ time, where n is the number of edges of the imprecise terrain.*

Note that Definition 10 can also be applied to non-regular terrains, since it is solely based on the potential watersheds of the imprecise minima. We can use the algorithm of Section 5.2 to compute proxies for these minima, and then use the algorithm of Lemma 18

to compute a fuzzy ridge between the watersheds of these proxies efficiently for non-regular terrains. However, note that the result may not be exactly the same as the fuzzy ridge according to Definition 10, because on a non-regular terrain, the potential watersheds of the proxies may be smaller than the potential watersheds of the imprecise minima.

6 Conclusions

In this paper we studied flow computations on imprecise terrains under two general models of water flow. For the surface model, where flow paths are traced across the surface of an imprecise polyhedral terrain, we showed NP-hardness for deciding whether water can flow between two points. For the network model, where flow paths are traced along the edges of an imprecise graph, we gave efficient algorithms to compute potential (maximal) and persistent (minimal) watersheds and potential downstream areas. Our algorithms also work for sets of nodes and can therefore be applied to reason about watersheds of areas, such as lakes and river beds.

In order to enable several extensions to these results in the network model, we introduced a certain class of imprecise terrains, which we call regular. We first defined when a set of nodes in an imprecise terrain can be considered a 'stable' imprecise minimum. We then described how to turn a non-regular terrain into a regular terrain using an algorithm by Gray et al. [12] and showed that this regularization algorithm preserves these imprecise minima. Interestingly, this algorithm also minimizes the number of minima of the terrain, while respecting the elevation bounds, as shown in [12].

We showed that persistent watersheds are nested on regular terrains and that these terrains have a fuzzy ridge structure which delineates the persistent watersheds of these stable minima. We gave an algorithm to compute this structure in $O(n \log n)$ time, where n is the number of edges of the terrain. The correspondence between the imprecise minima of the regular and the non-regular terrain suggests that this fuzzy watershed decomposition on the regular terrain also allows us to reason about the structure of the watersheds on the original non-regular terrain. We think that, even though our work is motivated by geographical applications, the results will be useful in other application areas where watersheds are being computed, for instance in image segmentation [24].

There are many open problems for further research.

Clearly, the contrast between the results in the surface model vs. the results in the network model leaves room for further questions, e.g., can we develop a model to measure the quality of approximations of water flow in the surface model, and how does it relate to the network model?

Surprisingly, a persistent watershed according to our current definition may consist of multiple connected components: the persistent watershed of a node q may contain a node p such that one cannot walk from p to q without leaving the watershed (see Appendix B). It can be debated whether this is an acceptable and possibly rare consequence of a sensible definition, or if this indicates that our definition needs to be refined or corrected.

Other flow models have been proposed in the GIS literature, e.g. D-∞, in which the

incoming water at a node is distributed among the outgoing descent edges according to steepness. These models can be seen as modified network models which approximate the steepest-descent direction more truthfully. In order to apply the techniques we developed for watersheds, we first need to formalize *to which extent* a node is part of a watershed in these models.

Acknowledgments. We thank Chris Gray for many interesting and useful discussions on the topic of this paper.

References

[1] K. Beven. *Environmental modelling: An uncertain future?* Routledge, 2009.

[2] M. Borga, E. Gaume, J. Creutin, and L. Marchi. Surveying flash floods: gauging the ungauged extremes. *Hydrol. Process.*, 22:3883–3885, 2008.

[3] W. Buytaert, D. Reusser, S. Krause, and J. Renaud. Why can't we do better than Topmodel? *Hydrol. Process.*, 22:4175–4179, 2008.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press and McGraw-Hill Book Company, third edition, 2009.

[5] W. Craddock, E. Kirby, N. Harkins, H. Zhang, X. Shi, and J. Liu. Rapid fluvial incision along the Yellow River during headward basin integration. *Nat. Geosci.*, 3:209–213, 2010.

[6] A. Danner, T. Mølhave, K. Yi, P. K. Agarwal, L. Arge, and H. Mitásová. TerraStream: from elevation data to watershed hierarchies. In *Proc. 15th ACM Int. Symp. on Geographic Information Systems (ACM-GIS 2007)*, pages 212–219, 2007.

[7] M. de Berg, P. Bose, K. Dobrint, M. J. van Kreveld, M. H. Overmars, M. de Groot, T. Roos, J. Snoeyink, and S. Yu. The complexity of rivers in triangulated terrains. In *Proc. 8th Canad. Conf. Comput. Geom.*, pages 325–330, 1996.

[8] M. de Berg, O. Cheong, H. Haverkort, J.-G. Lim, and L. Toma. The complexity of flow on fat terrains and its I/O-efficient computation. *Comput. Geom. Theory Appl.*, 43(4):331–356, 2010.

[9] M. de Berg, H. Haverkort, and C. Tsirogiannis. Implicit flow routing on terrains with applications to surface networks and drainage structures. In *Proc. 22nd ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 285–296, 2011.

[10] M. de Berg and C. P. Tsirogiannis. Exact and approximate computations of watersheds on triangulated terrains. In *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. GIS*, pages 74–83, 2011.

[11] P. F. Fisher and N. J. Tate. Causes and consequences of error in digital elevation models. *Prog. Phys. Geog.*, 30(4):467–489, 2006.

[12] C. Gray, F. Kammer, M. Löffler, and R. I. Silveira. Removing local extrema from imprecise terrains. *Comput. Geom. Theory Appl.*, 45(7):334–349, 2012.

[13] C. Gray, M. Löffler, and R. I. Silveira. Smoothing imprecise 1.5D terrains. *Int. J. Comput. Geometry Appl.*, 20(4):381–414, 2010.

[14] H. Haverkort and C. Tsirogiannis. Flow on noisy terrains: An experimental evaluation. In *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. GIS*, pages 84–91, 2011.

[15] F. Hebeler and R. Purves. The influence of elevation uncertainty on derivation of topographic indices. *Geomorphology*, 111(1-2):4–16, 2009.

[16] M. Henzinger, P. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.*, 55(1):3–23, 1997.

[17] Y. Kholondyrev and W. Evans. Optimistic and pessimistic shortest paths on uncertain terrains. In *Proc. 19th Canad. Conf. Comput. Geom.*, pages 197–200, 2007.

[18] R. D. Koster, S. P. P. Mahanama, B. Livneh, D. P. Lettenmaier, and R. H. Reichle. Skill in streamflow forecasts derived from large-scale estimates of soil moisture and snow. *Nat. Geosci.*, 3(9):613–616, 2010.

[19] J. Lindsay and M. Evans. The influence of elevation error on the morphometrics of channel networks extracted from DEMs and the implications for hydrological modelling. *Hydrol. Process.*, 22(11):1588–1603, 2008.

[20] Y. Liu and J. Snoeyink. Flooding triangulated terrain. In *Proc. 11th Int. Symp. Spatial Data Handling*, pages 137–148, Berlin, 2005.

[21] M. Mcallister and J. Snoeyink. Extracting consistent watersheds from digital river and elevation data. In *Proc. ASPRS/ACSM Annu. Conf*, 1999.

[22] A. Montanari. What do we mean by 'uncertainty'? The need for a consistent wording about uncertainty assessment in hydrology. *Hydrol. Process.*, 21:841–845, 2006.

[23] J. O'Callaghan and D. Mark. The extraction of drainage networks from digital elevation data. *Comput. Vision Graphics Image Process.*, 28(3):323–344, 1984.

[24] D. L. Pham, C. Xu, and J. L. Prince. Current methods in medical image segmentation. *Biomed. Eng.*, 2:315–337, 2000.

[25] B. Sivakumar. The more things change, the more they stay the same: the state of hydrologic modelling. *Hydrol. Process.*, 22:4333–4337, 2008.

[26] D. Tarboton. A new method for the determination of flow directions and upslope areas in grid dig. elev. models. *Water Resour. Res.*, 33(2):309–319, 1997.

[27] D. Tetzlaff, J. McDonnell, S. Uhlenbrook, K. McGuire, P. Bogaart, F. Naef, A. Baird, S. Dunn, and C. Soulsby. Conceptualizing catchment processes: simply too complex? *Hydrol. Process.*, 22:1727–1730, 2008.

[28] J. A. Vrugt, C. G. H. Diks, H. V. Gupta, W. Bouten, and J. M. Verstraten. Improved treatment of uncertainty in hydrologic modeling: Combining the strengths of global optimization and data assimilation. *Water Resour. Res.*, 41, 2005.

[29] S. P. Wechsler. Uncertainties associated with digital elevation models for hydrologic applications: a review. *Hydrol. Earth Syst. Sc.*, 11(4):1481–1500, 2007.

A Computing potential watersheds in linear time

Theorem 3. *The canonical realization of the potential watershed of a set of cells Q in an imprecise grid terrain of n cells can be computed in $O(n)$ time.*

Proof. The computation of potential watersheds in Section 4.1 has much in common with computing single-source shortest paths. In both cases, the goal is to compute a label $\delta(v)$ for each node v: in the case of potential watersheds it is the lowest elevation such that a flow path to a given destination q exists; in the case of shortest paths it is the distance from the given source q. During the computation, we maintain *tentative* labels $d[v]$ for each node v which are upper bounds on the labels to be computed. (The tentative label of a node that has not been discovered yet would be ∞.) The computations consist of a sequence of edge *relaxations*: when relaxing a directed edge (u, v), we try to improve (that is, lower) $d[v]$ based on the current value of $d[u]$, which is an upper bound on $\delta(u)$. Both problems share some crucial properties: for every node v that can be reached, there is a "shortest" path $\pi(v) = u_0, u_1, ..., u_k$ where $u_0 = q$ and $u_k = v$, the correct labels $\delta(u_0), \delta(u_1), ..., \delta(u_k)$ form a non-decreasing sequence, and when the edges on this path are relaxed in order from (u_0, u_1) to (u_{k-1}, u_k), the relaxation of (u_{i-1}, u_i) will correctly set $d[u_i]$ equal to $\delta(u_i)$. All that is necessary to compute all labels is that the sequence ρ of relaxations performed by the algorithm contains $\pi(v)$ as a subsequence, for each v. Note that the edges of $\pi(v)$ do not need to be consecutive in ρ: the labels along $\pi(v)$ are computed correctly even if the relaxations of $\pi(v)$ are interleaved with relaxations of other edges, or even with out-of-order relaxations of edges of $\pi(v)$.

There are several algorithms to find a sequence of relaxations ρ in the above setting, such that for every node v, the sequence ρ contains the relaxations of a shortest path $\pi(v)$ as a subsequence. These algorithms are usually known as algorithms to compute (single-source) shortest paths, but they can also be applied directly to the more general setting described above. Dijkstra's algorithm finds a sequence of relaxations that is optimal in the sense that it relaxes each edge only once. However, to achieve this, the algorithm needs $\Theta(n)$ operations on a priority queue of size $\Theta(n)$ in the worst case, where n is the number of nodes and edges in the graph [4].

An alternative is the algorithm of Henzinger et al. [16]. This algorithm uses a hierarchy of priority queues. Most priority queue operations in this algorithm are on small priority queues. The algorithm needs more relaxations than Dijkstra's algorithm, but still not more than $O(n)$. Provided the relaxations take constant time each, the whole algorithm runs in $O(n)$ time. However, the algorithm by Henzinger et al. only works if a recursive decomposition of the graph is provided that satisfies certain properties. Fortunately such

decompositions can be found in $O(n)$ time for planar graphs, and also for certain other types of graphs. In particular, it is easy to construct such a decomposition for a graph that represents a grid terrain model, even in the model where each cell can drain to one or more of its eight neighbors, for which the adjacency graph is non-planar. Let $r_1 < r_2 < ...$ be a sequence of powers of four. Now we can easily make a decomposition of the graph into square regions of $\sqrt{r_1} \times \sqrt{r_1}$ nodes; we group these together into regions of $\sqrt{r_2} \times \sqrt{r_2}$ regions, etc., generally grouping regions of $\sqrt{r_i} \times \sqrt{r_i}$ nodes into regions of $\sqrt{r_{i+1}} \times \sqrt{r_{i+1}}$ nodes (some regions at the boundary of the whole input grid may be slightly smaller). On each level i, the regions have size $\Theta(r_i)$ and each region has $\Theta(\sqrt{r_i})$ nodes on its boundary, thus each level forms a so-called r_i-division. We choose the region sizes such that they satisfy Equation (19) from Henzinger et al.

With this decomposition, the structure of the single-source shortest paths algorithm from Henzinger et al. can also be applied to the computation of potential watersheds on grid terrains. For grid terrains, $d_{\max} = O(1)$, and thus, the computation of the slope diagrams and the $O(n)$ relaxation steps from the "shortest-paths" algorithm take only $O(n)$ time. Together with $O(n)$ time for priority queue operations, we get a total running time of $O(n)$. □

B Persistent watersheds with multiple connected components

Lemma 21. *There exists a regular terrain that contains a persistent watershed that consists of more than one connected component.*

Proof. Refer to Figure 13. The figure shows five nodes with their elevation intervals. The edges (a, b), (b, d) and (c, d) have length 1. The edge (d, e) has length 1.6. From a and e, very steep edges lead downwards to nodes not shown in the figure. The potential watershed $\mathcal{W}_\cup(e)$ of e is $\{c, d, e\}$. The node d is not in the persistent watershed of e: if d has elevation more than $6\frac{1}{3}$, the flow path from d will lead to b, outside $\mathcal{W}_\cup(e)$. In that case c is a local minimum inside $\mathcal{W}_\cup(e)$. Whenever c is not a local minimum, the elevation of d must be less than 4, and the flow path from c will lead to d and on to e. Thus c is in the persistent watershed $\mathcal{W}_\cap(e)$ of e, but d is not, so we have $\mathcal{W}_\cap(e) = \{c, e\}$. □

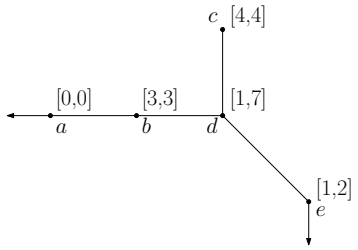


Figure 13: Example of disconnected persistent watershed on a regular terrain.